OBC INFORMATIQUE

N°82

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE

Le Centre mondial informatique

Le Discovery 1 de Opus

Test en cascade

Ports périphériques du C64

EDITIONS ATLAS

M 6062-82-12,50 F 90 FB - 3,80 FS - $ 1.95

Page manquante
(publicité et colophon)

Des projets ambitieux

La diffusion de la culture informatique, l'informatisation de l'univers individuel, telles sont quelques-unes des missions que s'est proposé le Centre mondial informatique.

Beaucoup de monde vient chaque jour au Centre mondial de l'informatique. Il faut dire que les objectifs du Centre sont vastes, depuis la volonté d'être un carrefour d'idées et de connaissances du monde entier en micro-informatique, jusqu'à la conception d'un micro bon marché. Mais ce dernier se fait toujours attendre.
(Cl. J.M. Voge/C.M.I.)

A deux pas des Champs-Élysées, à Paris, une façade de pierre taillée et de verre fumé abrite un lieu de vie, d'intérêt, d'apprentissage, de perfectionnement : le Centre mondial informatique et ressource humaine (C.M.I.). Ouvert sept jours sur sept, cet organisme « appartient à tous ceux dont la conscience sociale leur donne comme motivation première de s'unir à ceux qui veulent frayer le passage de l'ère industrielle à l'ère informatisée ».

Le rôle du Centre mondial est avant tout celui d'un lieu de vulgarisation de la « culture informatique ». Selon Jean-Jacques Servan-Schreiber, grand amateur de prospective industrielle, son créateur et président jusqu'en 1985, « les recherches et expériences du centre doivent contribuer à l'avènement du véritable ordinateur personnalisé avec lequel chaque citoyen pourra, progressivement, dialoguer, développer ses connaissances, transformer ses intuitions en création, de manière familière, s'approchant du dialogue entre êtres humains ».

Chacun, quels que soient ses connaissances, son milieu social, son âge — mais le C.M.I. est plus particulièrement destiné aux jeunes —, peut librement pousser la porte du 22, avenue Matignon. Là, un animateur se chargera de guider ses premiers pas et de favoriser le contact avec l'informatique. Le visiteur pourra lui-même s'asseoir devant une machine, taper ses premiers programmes au clavier, choisir un logiciel...

Un peu d'histoire

En novembre 1981, François Mitterrand annonçait la création du Centre mondial. Constitué en association loi de 1901 — mais son statut est actuellement en cours de transformation en établissement à caractère industriel et commercial —, il est officiellement fondé en février 1982. Placé sous la tutelle du ministère de la Recherche et de l'Industrie, et financé par des fonds publics, sa mission première est, d'une part, d'inciter la communauté scientifique et industrielle française à mettre au point un ordinateur personnel performant à moins de 1 000 F et, d'autre part, d'organiser des expériences d'intégration sociale de la micro-informatique.

En mars 1982, sous l'impulsion de J.J.S.S., les premiers micro-ordinateurs font leur entrée sous les lambris de l'immeuble de l'avenue Matignon. A l'époque, le responsable scientifique du Centre s'appelle Seymour Papert, le très réputé père de LOGO et chercheur en intelligence artificielle. Ce professeur de mathématiques et de sciences de l'éducation est arrivé du MIT (Massachusetts

Seymour Papert, le père du langage LOGO, fut le responsable scientifique du C.M.I. pendant un an. (Cl. Flammarion.)

Institute of Technology) avec son collègue Nicholas Negroponte, docteur en informatique, qui assure les fonctions de directeur du centre.

Malheureusement, les relations entre J.J.S.S. et Seymour Papert se détériorent rapidement, si bien que ce dernier quitte le C.M.I. à la fin de 1982. Negroponte en fait autant à la fin de son contrat. Une des raisons en est probablement le transfert de tutelle du centre qui, à l'initiative de J.J.S.S., est rattaché au ministère des P.T.T. Le président du C.M.I. justifie ce transfert par « la nécessité, pour un ordinateur personnel, de pouvoir être relié aux grandes banques de données internationales ». En effet, ces liaisons s'effectuent par câble, satellite ou téléphone, autant de technologies qui dépendent du ministère des P.T.T.

Pour faire face à ce double départ qui l'affaiblit et porte un coup sévère à sa crédibilité sur le plan international, le C.M.I. se structure et organise ses activités autour de quelques axes essentiels : les programmes de formation et de culture informatique, les « industries du savoir » (médecine et agriculture) et des activités internationales orientées principalement vers les pays du tiers monde.

De très nombreux jeunes profitent des micros mis à leur disposition pour se familiariser avec les techniques de programmation. (Cl. Service Audiovisuel/C.M.I.)

Entre-temps, le Centre change une nouvelle fois de tutelle, puisqu'il est, depuis 1984, rattaché au ministère du Plan et de l'Aménagement du territoire, ouvrant ainsi une nouvelle voie à une interaction dynamique et, par toutes les structures officielles, avec l'ensemble des ministères, administrations et collectivités concernés.

Ainsi, le programme 1985 du Centre fait apparaître trois axes opérationnels :

- La mise au point et l'évaluation d'outils d'informatisation.
- La capacité de présence immédiate pour répondre à des besoins sociaux urgents.
- L'échange permanent avec les pôles mondiaux où naît la science neuve, essentielle, de l'application de l'informatique personnelle aux capacités de chacun.

La capacité d'action immédiate du C.M.I. pour répondre à l'éclosion de besoins pressants et nouveaux, dus à la grande mutation sociale, est illustrée par trois exemples récents :

- La création de centres technologiques régionaux, appelés, depuis l'été 1984, à démultiplier les actions du Centre de Paris, et celle d'ateliers de pratique informatique progressivement destinés à toutes les collectivités locales.
- La création et la mise en service, à l'occasion de la crise de la sidérurgie, du Groupe de modernisation de la Lorraine, à l'initiative des pouvoirs régionaux.
- La création du Centre de formation technologique des travailleurs de l'automobile, afin de préparer tous les travailleurs à la robotisation des usines.

Le Centre intervient dans la phase initiale des projets. Il contribue à définir les objectifs, évaluer les besoins, assembler les moyens, en particulier les hommes, orienter les actions vers des secteurs et des métiers d'avenir. Son rôle est, en règle générale, celui d'un catalyseur.

L'irrigation par la culture informatique

La société moderne entre dans l'ère de l'informatique. L'ambition du Centre est d'y préparer chacun, afin d'éviter le chômage que pourrait créer l'informatisation et la robotisation de tous les secteurs du marché du travail. Il est donc nécessaire que les jeunes en particulier se familiarisent avec cette « deuxième langue maternelle » afin de bien s'insérer dans le monde de demain. C'est dans cette optique qu'a été lancée l'opération « 250 000 micros » qui doit permettre d'équiper écoles primaires, collèges, lycées, collectivités, ateliers de pratique informatique.

Il ne s'agit pas de former des informaticiens. Il ne s'agit pas de former à l'informatique, mais, par des outils informatiques, vers l'ensemble des métiers. L'exemple actuel de l'antenne du Centre mondial à Marseille le montre : la formation informatique y est donnée pour les métiers de plombier, menuisier, réparateur, électricien, etc. Les délais de formation sont bien plus courts, et la qualité est très supérieure.

Jean-Jacques Servan Schreiber eut l'idée du C.M.I. et en fut le responsable jusqu'à il y a peu de temps.
(Cl. J.M. Voge/C.M.I.)

Les industries du savoir

Deux domaines sont abordés par le C.M.I. : la médecine et l'agriculture.

Le projet « Médecine » organise, autour du Centre mondial, l'ensemble de la recherche et des travaux de six centres hospitalo-universitaires à Paris, Marseille, Montpellier, Bordeaux et Lille. L'objectif général est de développer des méthodes de traitement avancé de l'information médicale. Il s'agit, d'une part, de l'intelligence artificielle (systèmes experts, bases de données de 3e génération, langage naturel); d'autre part, de mathématique appliquée (modélisation, traitement du signal, reconnaissance des formes pour réaliser des systèmes d'aide à la décision médicale).

Ces méthodes seront évaluées, puis industrialisées et diffusées. Pour mener à bien ces projets, le Centre mondial fait équiper les cinq centres médico-informatiques participant à la coordination du projet. Leur interconnexion est également réalisée par le centre, qui prend en charge la formation des nouveaux chercheurs spécialisés en intelligence artificielle appliquée à la médecine.

Dans le domaine agricole, le C.M.I. a démarré trois programmes en France : deux de formation et un de recherche appliquée. Pour aider les centres de formation agricole, les C.F.P.P.A. (Centres de formation professionnelle pour agriculteurs) vont tous être équipés de micro-ordinateurs, et interconnectés. Autre domaine clé : la gestion informatisée des entreprises agricoles. Pour cela, 30 000 agriculteurs et conseillers seront formés sans délai. Par ailleurs, l'approche de l'intelligence artificielle dans des systèmes experts est particulièrement adaptée au développement de l'agriculture intensive. Elle permet une interaction homme-machine en langage quasi naturel, et une approche de conseil de premier plan pour tout agriculteur.

Les programmes internationaux

Le troisième volet des activités du C.M.I. concerne les relations avec les pays du tiers monde, en particulier dans le domaine de la santé; ce domaine complexe inclut nutrition, alphabétisation, niveau d'hygiène général, aussi bien que prévention, vaccination et médecine de soins.

Le micro-ordinateur de brousse constitue un exemple d'utilisation de l'informatique en environnement pauvre. L'équipe santé-informatique s'est en effet attachée à trouver un outil (le micro « Husky ») portable, autonome, étanche et robuste, qui fournira à l'infirmier isolé, dans la pratique quotidienne, à la fois une aide au diagnostic et une aide à la thérapeutique.

D'autres actions sont envisagées, notamment dans le domaine de l'enseignement du secourisme, de l'hygiène, etc., faisant appel au vidéodisque interactif.

En agriculture, le C.M.I. a constitué une équipe en Côte-d'Ivoire pour développer et tester un vidéodisque interactif sur les pratiques culturales de base en climat tropical. Des systèmes experts devraient permettre d'améliorer les productions des pays les moins avancés.

Un projet de diffusion de la culture informatique est mené en collaboration avec le ministère de la Recherche scientifique et technique du Sénégal, ainsi qu'avec l'École normale supérieure de Dakar. Il s'agit d'un projet pilote visant à former des formateurs à l'utilisation de la micro-informatique, au service des enfants, des maîtres et des adultes.

Enfin, des stages internationaux, dont le premier a réuni à Paris, en novembre-décembre 1983, trente représentants de quinze pays différents, permettent aux participants de s'informer sur les ressources qu'apporte l'informatique, d'acquérir de nouvelles connaissances dans ce domaine, d'élaborer un projet de diffusion de la culture informatique propre à leur pays, et d'établir un réseau de communications et d'échanges.

Le C.M.I. travaille également en direction du tiers monde et aide à la création d'autres centres, comme au Sénégal.
(Cl. Service Audiovisuel/C.M.I.)

Un homme à la mer

Notre voyage sera marqué par une série d'événements imprévus. Chaque semaine, il pourra s'en produire deux.

Afin de pouvoir prendre en compte, au sein de notre jeu, les difficultés et les contingences réellement rencontrées par les marins du XVI[e] siècle, nous devrons permettre au programme de faire intervenir, au besoin, divers événements imprévus. Certains peuvent se révéler bénéfiques — un vent favorable, par exemple — mais d'autres auront de lourdes conséquences : la perte de provisions ou d'hommes d'équipage, par exemple.

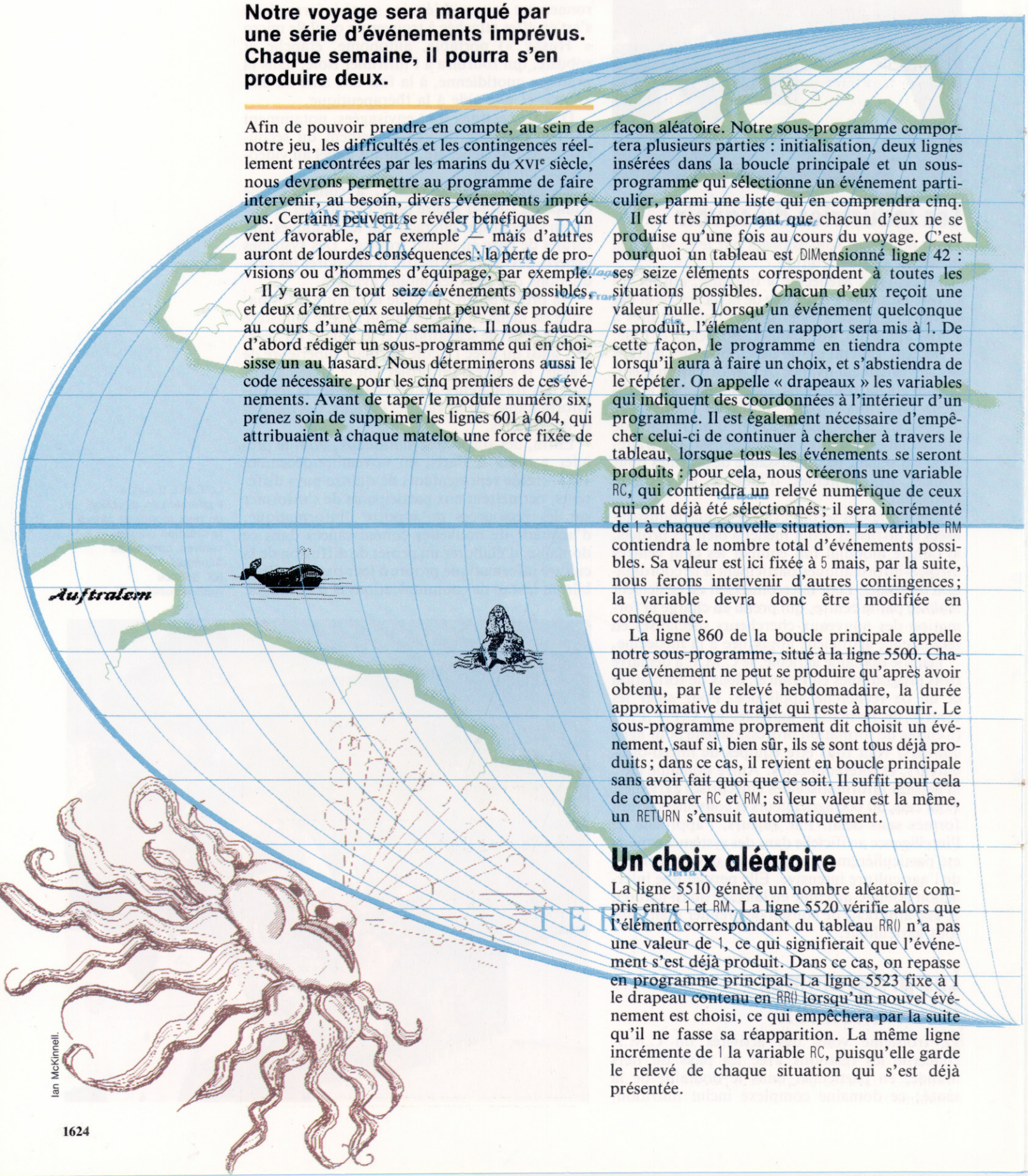
Il y aura en tout seize événements possibles, et deux d'entre eux seulement peuvent se produire au cours d'une même semaine. Il nous faudra d'abord rédiger un sous-programme qui en choisisse un au hasard. Nous déterminerons aussi le code nécessaire pour les cinq premiers de ces événements. Avant de taper le module numéro six, prenez soin de supprimer les lignes 601 à 604, qui attribuaient à chaque matelot une force fixée de façon aléatoire. Notre sous-programme comportera plusieurs parties : initialisation, deux lignes insérées dans la boucle principale et un sous-programme qui sélectionne un événement particulier, parmi une liste qui en comprendra cinq.

Il est très important que chacun d'eux ne se produise qu'une fois au cours du voyage. C'est pourquoi un tableau est DIMensionné ligne 42 : ses seize éléments correspondent à toutes les situations possibles. Chacun d'eux reçoit une valeur nulle. Lorsqu'un événement quelconque se produit, l'élément en rapport sera mis à 1. De cette façon, le programme en tiendra compte lorsqu'il aura à faire un choix, et s'abstiendra de le répéter. On appelle « drapeaux » les variables qui indiquent des coordonnées à l'intérieur d'un programme. Il est également nécessaire d'empêcher celui-ci de continuer à chercher à travers le tableau, lorsque tous les événements se seront produits : pour cela, nous créerons une variable RC, qui contiendra un relevé numérique de ceux qui ont déjà été sélectionnés ; il sera incrémenté de 1 à chaque nouvelle situation. La variable RM contiendra le nombre total d'événements possibles. Sa valeur est ici fixée à 5 mais, par la suite, nous ferons intervenir d'autres contingences ; la variable devra donc être modifiée en conséquence.

La ligne 860 de la boucle principale appelle notre sous-programme, situé à la ligne 5500. Chaque événement ne peut se produire qu'après avoir obtenu, par le relevé hebdomadaire, la durée approximative du trajet qui reste à parcourir. Le sous-programme proprement dit choisit un événement, sauf si, bien sûr, ils se sont tous déjà produits ; dans ce cas, il revient en boucle principale sans avoir fait quoi que ce soit. Il suffit pour cela de comparer RC et RM ; si leur valeur est la même, un RETURN s'ensuit automatiquement.

Un choix aléatoire

La ligne 5510 génère un nombre aléatoire compris entre 1 et RM. La ligne 5520 vérifie alors que l'élément correspondant du tableau RR() n'a pas une valeur de 1, ce qui signifierait que l'événement s'est déjà produit. Dans ce cas, on repasse en programme principal. La ligne 5523 fixe à 1 le drapeau contenu en RR() lorsqu'un nouvel événement est choisi, ce qui empêchera par la suite qu'il ne fasse sa réapparition. La même ligne incrémente de 1 la variable RC, puisqu'elle garde le relevé de chaque situation qui s'est déjà présentée.



Ian McKinnell.

A la ligne 5525, une instruction ON X GOTO envoie le programme en direction de l'événement sélectionné. ON...GOTO et ON...GOSUB permettent d'éviter un emploi excessif d'instructions IF...THEN. Le principe en est simple, puisque chaque ON est suivi de plusieurs numéros de ligne. L'une d'elles est choisie selon la valeur de la variable : la première ligne si X a pour valeur 1, la deuxième si c'est 2, et ainsi de suite. Prenez bien garde si vous possédez un Spectrum, dont le BASIC ne comporte pas cette instruction. Reportez-vous au tableau « Variantes de basic ».

Il y a cinq instructions, bien qu'il n'y ait en fait que deux événements. Le premier est la perte d'un homme d'équipage, passé par-dessus bord. Le second, c'est la disparition d'une partie des provisions. La ligne 5540 fait mourir le malheureux. Les lignes 5570 à 5595 gèrent la perte de la nourriture. On remarquera que l'instruction ON X GOTO se termine par quatre numéros de ligne identiques : si le nombre aléatoire est compris entre 2 et 5 (donc quatre possibilités différentes), on passera à chaque fois en 5540.

En cas de disparition d'un matelot, la ligne 5548 provoque l'affichage d'un message qui nous en informe. La ligne 5554 précise combien il reste désormais d'hommes d'équipage en soustrayant 1 de CN (nombre total de marins). Toutefois, la valeur de CN n'est pas encore modifiée ; ce sera fait plus tard, lors du relevé hebdomadaire. Le programme vérifiera à ce moment si, par hasard, la force de tel ou tel n'a pas été fixée à -999, ce qui indique qu'il est mort.

Il faut naturellement faire le choix d'une victime. C'est la tâche de la ligne 5558, qui parcourt le tableau consacré à l'équipage, ignore toutes les valeurs égales à zéro ou à -999, et sélectionne le premier matelot vivant qu'elle rencontre. Le malheureux est alors jeté à la mer par la ligne 5562, qui fixe sa force à -999. T se voit alors affecter une valeur de 16 de façon que le programme quitte la boucle.

Les quatre autres événements — de 2 à 5 dans le tableau RR — concernent la perte de provisions. Chacun d'eux concerne un type de provisions particulier. Un petit raffinement : la quantité de provisions anéanties est elle-même fixée de façon aléatoire. PA(1), PA(2), PA(3) et PA(4) correspondent respectivement aux fruits, aux légumes, à la viande et à l'eau.

La ligne 5572 décrémente de 1 le nombre aléatoire X généré en 5510. En effet, il s'est déjà produit quelque chose : un matelot a disparu. X sera désormais compris entre 1 et 4, ce qui correspond aux quatre types de provisions. Si l'une d'entre elles est déjà épuisée, il ne se produira rien : c'est le rôle de la ligne 5574. Dans le cas contraire, le joueur est informé de ses pertes de nourriture.

Ce qui disparaît peut représenter un tiers, un quart ou un cinquième de la quantité totale. Cette quantité est contenue dans le tableau PA(). La ligne 5592 divise l'élément correspondant par trois, quatre ou cinq, et soustrait le résultat de la quantité originelle. La ligne 5594 permet alors de savoir combien ce qui reste pourra durer (en semaines).

Module six : événements imprévus

Initialisations variables

```
42 DIMRR(16)
43 REM INDICATEURS PERMETTANT DE SAVOIR SI UN EVENEMENT
ALEATOIRE S'EST PRODUIT
44 RC=0
45 REM NB D'EV. ALEA. JUSQUE LA
46 REM=5
47 REM NB D'EV. ALEA DANS LE PROGRAMME
```

Addition à la boucle principale

```
860 GOSUB5500
861 REM EVENEMENTS ALEATOIRES
```

S/P Evénements aléatoires

```
5500 REM GENERATEUR EV. ALEA.
5502 IFRC=RM THEN RETURN
5503 REM QUITTE SI TOUS LES EVENEMENTS SE SONT PRODUITS
5504 PRINTCHR$(147)
5505 GOSUB9200
5510 X=INT(RND(1)*RM)+1
5515 REM NB ALEATOIRE COMPRIS ENTRE 1 ET RM
5520 IFRR(X)=1 THEN RETURN
5522 REM RETURN SI DEJA SURVENU
5523 RR(X)=1:RC=RC+1
5524 REM INCREMENTE COMPTEUR EVENEMENTS ET PRECISE CE QUI
S'EST PRODUIT
5525 ON X GOTO 5540,5570,5570,5570,5570
5528 REM SELON EVENEMENTS CHOISI
5530 PRINT:S$=K$:GOSUB9100
5535 GETI$:IFI$=""THEN5535
5539 RETURN: REM RETOUR EN BOUCLE PRINCIPALE
5540 REM UN HOMME A LA MER !
5542 PRINT
5543 S$=" CETTE SEMAINE+":GOSUB9100
5545 PRINT:GOSUB9200
5546 S$=" UN HOMME A DISPARU EN MER+":GOSUB9100
5548 S$=" PENDANT UNE TEMPETE+":GOSUB9100
5550 PRINT:GOSUB9200
5552 S$="VOTRE EQUIPAGE EST REDUIT A+":GOSUB9100
5554 PRINTCN-1:"MEMBRES"
5558 FORT=1TO16
5559 REM MATELOT A SACRIFIER
5560 IFTS(T,2)=0ORTS(T,2)=-999THEN5566
5562 TS(T,2)=-999:REM TERMINE !
5564 T=16
5566 NEXT
5568 GOTO5530
5570 REM EVENEMENTS 2 A 5 - PROVISIONS PERDUES
5572 X=X-1:REM X CONCERNE LES PROVISIONS (1-4)
5574 IFPA(X)=0 OR PA(X)=-999 THEN 5530
5576 REM PAS DE CHANGEMENT SI PROVISIONS DEJA PERDUES
5578 PRINT
5580 S$=" PENDANT LA SEMAINE+":GOSUB9100
5582 PRINT:GOSUB9200
5584 PRINT "                              ":GOSUB9100
5586 S$="VOUS AVEZ PERDU EN MER":GOSUB9100
5588 PRINT:GOSUB9200
5590 S$="UNE PARTIE DE VOS PROVISIONS DE+":P$X:GOSUB9100
5592 PA(X)=PA(X)-INT(PA(X)/(INT(RND(1)*3)+3))
5593 REM "IL VOUS RESTE ENVIRON POUR"
5594 PRINTINT(PA(X)/(CN*PN(X)));
5595 PRINT "SEMAINES ";P$(X);
5599 GOTO5530
```

Variantes de basic

Spectrum :

Procédez aux modifications suivantes :

5504 CLS
5525 IF X=1 THEN GOTO 5540
5526 IF X>1 AND X<6 THEN GOTO 5570
5535 LET I$=INKEY$: IF $="" THEN GOTO 5535

BBC Micro :

Procédez aux modifications suivantes :

5504 CLS
5535 I$=GET$

Tableaux pascal

Le PASCAL utilise moins les tableaux qu'un autre langage. Quel usage en fait-il pour les groupements et les indices, et quelles sont les contraintes?

Les tableaux sont pour de nombreux langages un moyen simple pour ranger des données courantes telles que des listes, des tables et des matrices, dans une structure informatique de données. C'est également un moyen d'accès efficace aux éléments, en vue de leur traitement. L'utilisation presque universelle des tableaux n'a cependant rien à voir avec une simple habitude : le plus souvent, il n'existe pas d'autres méthodes de structuration des données dans les premiers langages. Le premier langage évolué, le FORTRAN, n'avait lui-même que des nombres complexes et des tableaux. Son descendant, le BASIC, a « oublié » les nombres complexes. C'est la raison pour laquelle la boucle FOR ... NEXT est la seule structure d'itération définie en BASIC.

Cela ne posait pas de problème lorsque le FORTRAN n'était utilisé que pour des opérations matricielles sur tous les éléments d'un tableau, et lorsque le BASIC ne servait qu'à initier les étudiants aux difficultés de la programmation en PASCAL. Nous avons déjà vu quelques exemples de la souplesse qu'apportent au PASCAL les boucles conditionnelles (WHILE et REPEAT). Peut-être plus important encore, nous commençons à réaliser la très grande puissance, la maniabilité et la facilité d'expression que permettent certaines structures de données PASCAL telles que les tableaux et les enregistrements. La suprématie des tableaux comme mode universel de programmation est donc considérablement réduite en PASCAL.

Souplesse des tableaux pascal

Bien que la plupart des programmeurs se sentent à l'aise avec les tableaux PASCAL, deux caractéristiques doivent être soulignées. Le PASCAL offrant un tel choix de méthodes de structuration des données, de nombreux problèmes de programmation seront résolus plus efficacement par d'autres constructions. Il faut savoir également que les tableaux PASCAL sont entièrement libres de structures internes pour leurs éléments, ce qui permet une plus grande souplesse.

La définition d'un tableau réserve de la place mémoire pour une certaine taille de tableau et indique à la fois le type de base de son indice (dimension) et le type de chaque élément. Ainsi par exemple :

```
TYPE
   CompteCar = ARRAY[«A» .. «Z»]OF entier;
VAR
   liste : CompteCar;
```

réserve de la mémoire pour vingt-six entiers référencés par l'identificateur du tableau suivi d'une expression d'indexation entre crochets. Les tableaux utilisent uniquement des crochets, comme le BASIC des origines (l'utilisation ultérieure de parenthèses était simplement destinée aux jeux de caractères sans crochets). Pour accéder à un entier déterminé dans la liste ci-dessus, nous pouvons écrire :

```
liste [«M»], ou liste [pred (symbole)]
```

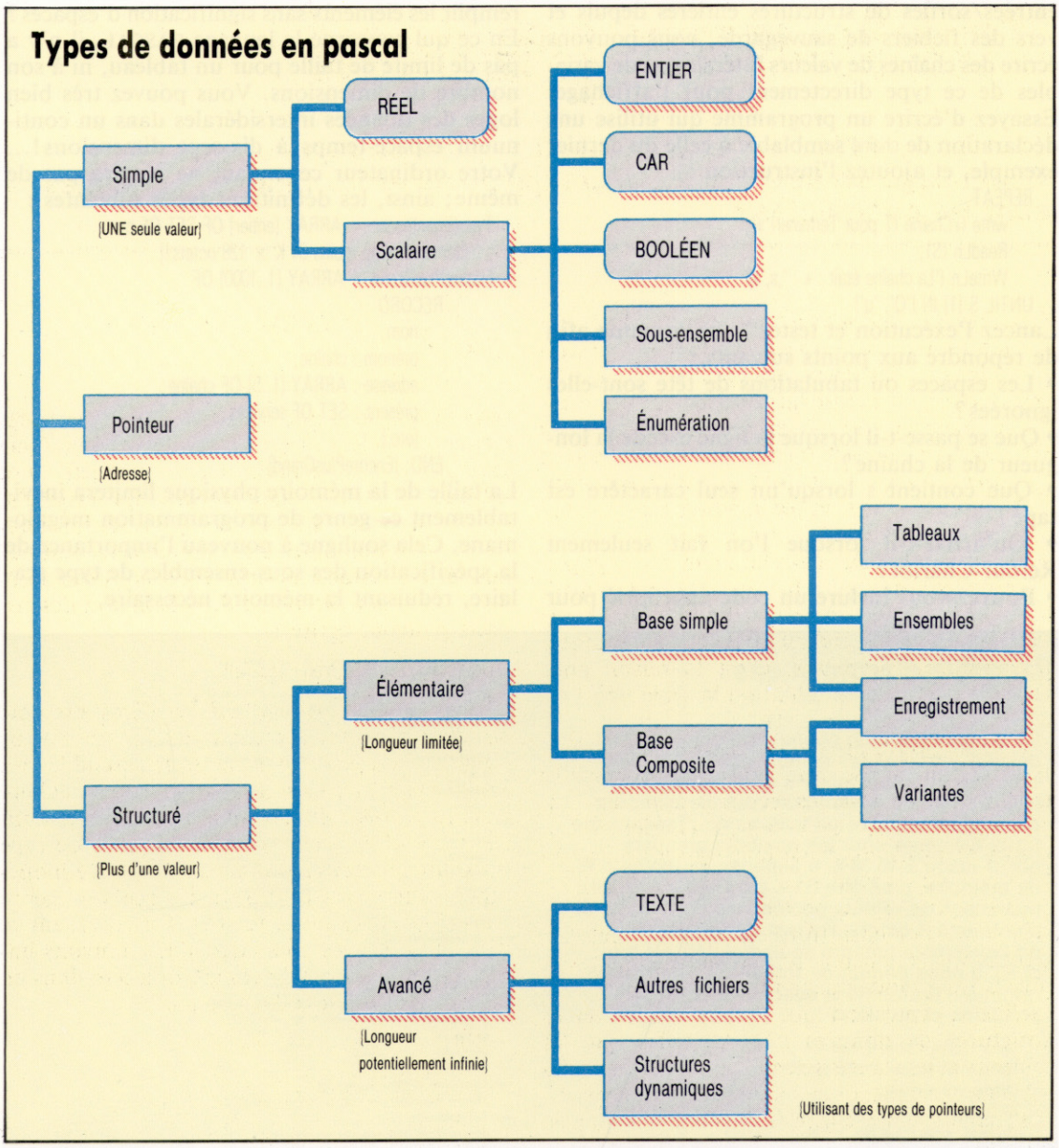
L'expression pred (symbole) du deuxième exemple doit bien sûr donner comme résultat une valeur pour car comprise dans le sous-ensemble A-Z. Toute valeur de type authentiquement scalaire peut être utilisée comme indice pour dimensionner un tableau, mais non des valeurs de type structuré ou réel. Cela permet de rendre inopérante toute tentative illogique d'accès au genre liste [«deuxième»] ou drapeau [3,75]. En reprenant la définition donnée plus haut, nous pouvons initialiser tout élément du compteur à zéro dans un tableau de type « CarCompte » par :

```
VAR
   lettre : car;
   compteur : CompteCar;
BEGIN
FOR lettre = 'A' TO 'Z' DO
   compteur [lettre] : = 0
```

et ainsi de suite. Manifestement, compteur [1] est illégal pour la syntaxe présente (mauvais type d'indice), et compteur [«a»] n'existe pas (la dimension excède le sous-ensemble défini). C'est pourquoi il est toujours conseillé de définir des identificateurs de types pour les variables d'indexation. Et de toute manière, nous verrons plus loin que ces derniers sont indispensables pour la définition de procédures et de fonctions utilisateur.

Lorsque ces sous-ensembles sont à leur tour définis sur des valeurs données dans une définition CONST, il se peut que l'on ait à réécrire la totalité du programme pour tenir compte de structures de différents formats dues à la modification d'une ligne. Par exemple, et bien qu'il n'existe pas de chaîne-type prédéclarée en PASCAL, on pourrait en façonner de la manière suivante (ce n'est pas nécessairement la meilleure) :

```
CONST
```



Typique
La structure de données très disciplinée du PASCAL oblige le programmeur à adopter une approche plus méthodique de construction d'un programme. Seuls deux des trois types de données de base montrés ici peuvent se subdiviser en sous-catégories. La multiplicité des types simples et structurés de données suppose des définitions rigoureuses de toutes les données du programme. Cet effort supplémentaire demandé au programmeur se verra récompensé par des solutions plus élégantes que celles des autres langages.
(Cl. Liz Dixon.)

```
    LongueurChaîne = 25;
TYPE
    FormatChaîne = 1 .. LongueurChaîne;
    chaîne = PACKED ARRAY [FormatChaîne]
    OF car;
```

Remarquez que le mot réservé PACKED (groupe) peut précéder tout mot réservé de type structuré (SET, ARRAY, RECORD ou FILE/ ensemble, tableau, enregistrement ou fichier). En groupant des structures de données, nous pouvons réaliser de substantielles économies de mémoire, spécialement avec les ordinateurs dont la taille des mots est longue (16/32 bits ou davantage).

La seule circonstance où vous devez faire attention en utilisant des groupes pour des tableaux concerne l'utilisation de constantes de chaînes littérales. Une chaîne de caractères placée entre guillemets est considérée comme indexée à partir de un (et non de zéro), et donc déclarée implicitement comme :

```
PACKED ARRAY [1..N] OF car
```

où N est le nombre de caractères de la chaîne.

Soit le programme :

```
PROGRAMME ChaîneGroupe;
CONST
    Pascal = 'Pascal';
TYPE
    chaîne = PACKED ARRAY [1..10] OF car;
VAR
    S : chaîne;
BEGIN
    S: = Pascal;
    S: = 'Trop de caractères';
    S: = 'Pascal'
END
```

Les deux instructions du corps du programme sont illégales du fait de leur incompatibilité de longueur. La troisième est acceptable (quatre espaces étant utilisés comme caractères de remplissage). Si la définition du type chaîne n'était pas groupée, toutes les assignations à des valeurs littérales seraient incompatibles. Ces restrictions ne constituent pas cependant un vrai problème dans la pratique. Bien que PASCAL n'utilise officiellement les procédures read et write que pour les

entrées/sorties de structures entières depuis et vers des fichiers de sauvegarde, nous pouvons écrire des chaînes de valeurs littérales ou de variables de ce type directement pour l'affichage. Essayez d'écrire un programme qui utilise une déclaration de chaîne semblable à celle du dernier exemple, et ajoutez l'instruction :

```
REPEAT
    write («Chaîne (T pour Terminer):»);
    ReadLn (S);
    WriteLn ('La chaîne était : «  ',s,'»)
UNTIL S [1] IN ['Q', 'q']
```

Lancez l'exécution et testez le programme afin de répondre aux points suivants :

- Les espaces ou tabulations de tête sont-elles ignorées ?
- Que se passe-t-il lorsque la ligne excède la longueur de la chaîne ?
- Que contient s lorsqu'un seul caractère est tapé ?
- Qu'arrive-t-il lorsque l'on fait seulement Retour-Chariot ?
- Pouvez-vous inclure un code approprié pour remplir les éléments sans signification d'espaces ?

En ce qui concerne le langage PASCAL, il n'y a pas de limite de taille pour un tableau, ni à son nombre de dimensions. Vous pouvez très bien loger des données intersidérales dans un continuum espace-temps à dix-sept dimensions !... Votre ordinateur cependant ne réagira pas de même ; ainsi, les définitions-types suivantes :

```
TypeGigantesque = ARRAY [entier] OF SET OF car;
    {demande d'au moins 64 K x 128 octets!}
EncorePlusGrand = ARRAY [1..1000] OF
        RECORD
            nom,
            prénom : chaîne;
            adresse : ARRAY [1..5] OF chaîne;
            présent : SET OF services;
            {etc.}
        END; {EncorePlusGrand}
```

La taille de la mémoire physique limitera inévitablement ce genre de programmation mégalomane. Cela souligne à nouveau l'importance de la spécification des sous-ensembles de type scalaire, réduisant la mémoire nécessaire.

Le crible d'Ératosthène

Ce programme bien connu des bancs d'essai, destiné à trouver les nombres premiers, est optimisé pour des raisons de rapidité d'exécution, en laissant de côté les nombres pairs et en utilisant un tableau de drapeaux (de 8 192 éléments pour les nombres premiers allant jusqu'à 16 384). Il suppose au moins 8 K de mémoire, peut-être 32 K pour les langages manquant de valeurs booléennes sur 1 octet et d'entiers à 4 octets. En PASCAL, la totalité de l'ensemble prendra seulement 16 384 bits (2 K). Il nous permet de formuler l'algorithme originel d'Ératosthène avec davantage de précision et de clarté :

```
Mettez tous les nombres (1..max) dans un tamis
Retirez-en le nombre un (premier par définition)
{Affichez-le si nécessaire}
REPEAT
    Retirez du lot le plus petit nombre
    {affichez ce nombre premier}
    retirez tous ses multiples
UNTIL il n'y a plus de nombres
```

Un ensemble aussi important ne sera permis qu'avec des gros ordinateurs, mais nous pouvons en simuler l'exploitation sur un jeu plus restreint de nombres (de 100 à 1 000 éléments selon votre compilateur). On obtient les indices de tableau pour l'ensemble à partir de la division entière du nombre. L'appartenance à l'ensemble est représentée par N modulo 100 ou 1 000 selon le cas. A moins que vous n'ayez un ordinateur à 100 bits, la vitesse d'exécution sera cependant plus lente qu'avec la version tableau.

```
PROGRAMME ListeCrible ( sorties );
{Trouver les nombres premiers selon l'algori-
thme d'Eratosthènes}
CONST
    TailleEnsemble = 1001   {Implémentation}
    TaillePrédéterminéeEnsemble = 99; {Tail-
     leEnsemble - 1}
    NombrePremierMax = 16383;   {Pour Entier-
    Max = 32767}
    ListeMax  =  163;  {NombrePremierMax  DIV
    TailleEnsemble}

TYPE
    FourchetteNombresPremiers = 1  ..  Nombre-
    PremierMax1
    dimension                 = 0  ..  ListeMax1
    FourchetteEnsemble        = 0  ..  TaillePré-
déterminéeEnsemble1
    Crible                    = SET OF FourchetteEnsemble1
    Eratosthène               = ARRAY  dimension  OF Cri-
    ble1
    cardinal                  = 0  ..  EntierMax1

VAR
    Crible                    : Eratosthène1
    compte,
    multiple                  : cardinal1
    index                     : dimensions1
    N                         : FourchetteNombresPremiers1
    Nombre                    : 0  ..  TaillePré-
déterminéeEnsemble1

BEGIN
    WriteLn1
    WriteLn  (  ' Crible 'Eratosthènes'   : 50)1
    WriteLn ( '==============' : 50 )1
    WriteLn1
    WriteLn1

FOR indice := 0 TO ListeMax DO
    Crible [indice] :=  [0  ..  TaillePré-
déterminéeEnsemble] 1
    mettre tous les nombres ensembles

Crible  [0]  :=  [2  ..  TaillePrédétermi-
néeEnsemble ]1
 WriteLn ( 1 )1   nombre premier par dé-
finition
 compte := 11

FOR N := 2 TO PremierMax DO
  BEGIN
     indice := N DIV TailleEnsemble1
     nombre := N MOD TailleEnsemble1

     IF nombre IN Crible [indice] THEN
        BEGIN
          compte  :=  suivant ( compte )1
         WriteLn ( N )1
          Crible [indice] := Crible [in-
          dice] - [nombre] 1
          multiple := N + N1

          WHILE multiple   =  PremierMax DO

           BEGIN
           indice  :=  multiple  DIV
           TailleEnsemble1
           Crible [ indice ] := Crible
           [ indice ]
           - [ multiple MOD TailleEn-
           semble ]1
           multiple  :=  multiple  + N
        END
     END
  END1
  WriteLn ( compte : 25, ' nombres pre-
  miers trouvés.')
END.
```

Test en cascade

Pour comprendre ce que signifie un test en cascade, nous allons dans ce premier exemple effectuer une comparaison entre une valeur variable et quelques valeurs constantes.

Un test en cascade consiste à établir une série de questions, continues et éliminatoires, qui ont comme but de comparer le contenu d'une variable avec celui de diverses valeurs constantes. A chaque question, on retrouve les deux issues possibles, oui et non.

Pour prendre un exemple extrêmement facile, cherchons à savoir quel jour de la semaine nous sommes ; nous nous interrogeons dans ce cas sur le numéro d'ordre que le jour occupe (figure 1). On peut observer que six questions seulement ont été employées. Question de logique, puisque dans cet exemple, la septième n'est pas nécessaire.

Poursuivons notre raisonnement par un autre exemple (figure 2). Supposons qu'une machine pèse des coussinets mélangés qui arrivent jusqu'à elle. Elle les pèse un par un. Si leur poids est de 10 g, elle les envoie au dépôt 1 ; s'ils pèsent 25 g, elle les envoie au dépôt 2, et si leur poids est de 50 g, elle les envoie au dépôt 3. Les coussinets dont le nombre de grammes ne correspond pas aux poids fixés au préalable sont rejetés et envoyés vers un conteneur spécial réservé aux produits défectueux.

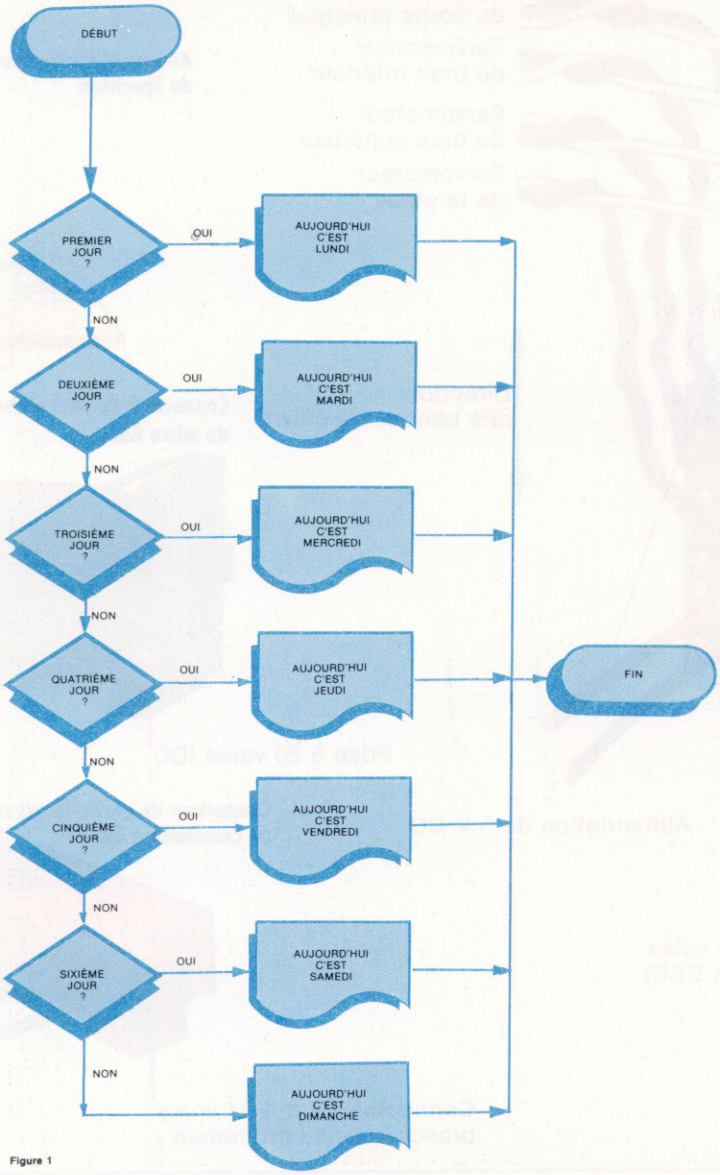


Figure 1

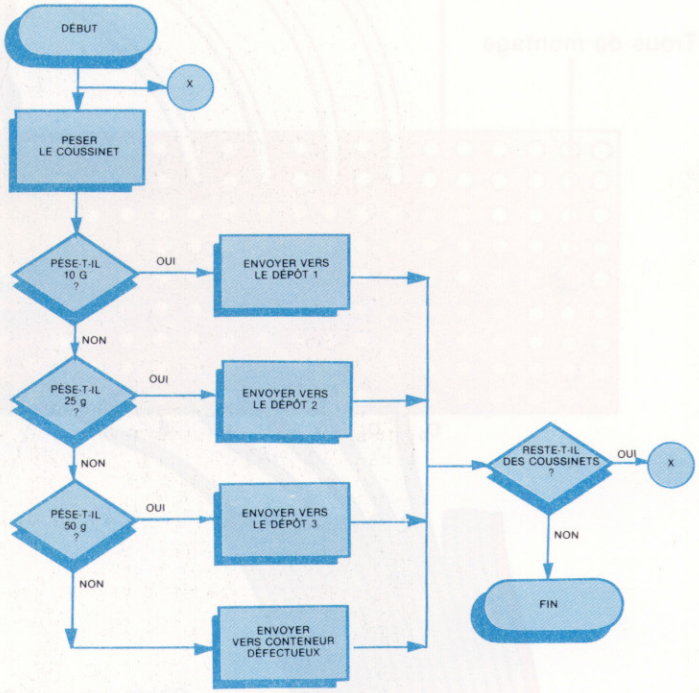
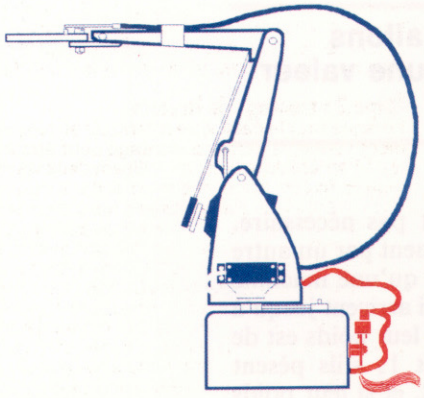


Figure 2

Entièrement armé

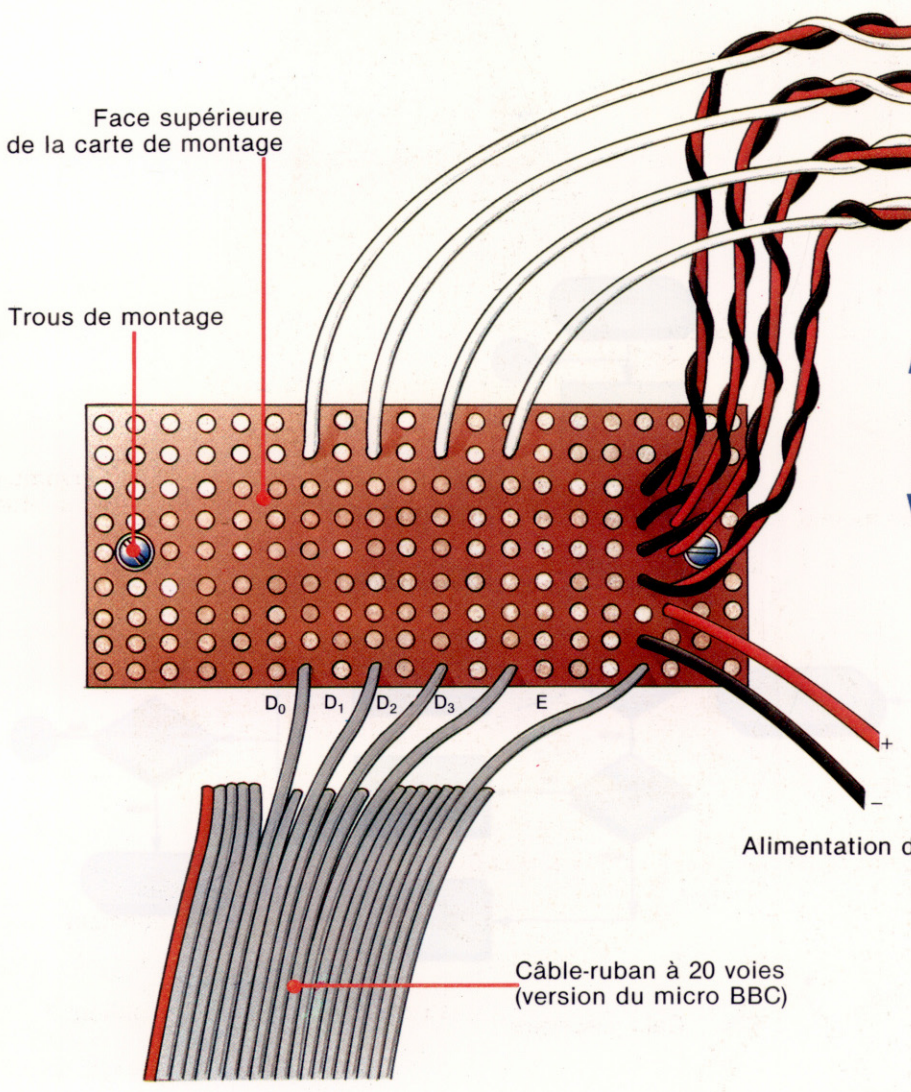


Il ne reste plus qu'à effectuer les connexions électriques pour terminer notre projet de robot. L'alimentation et les lignes de commande arrivent.

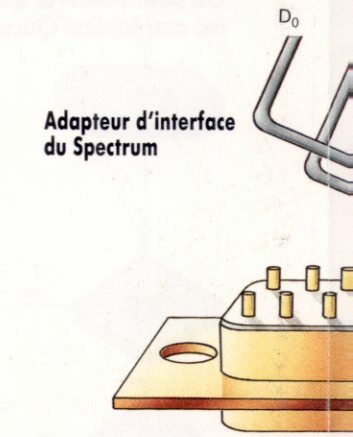
Étape 1 : les connexions de la carte de montage
Trois fils sont connectés à chaque servomoteur. Sur les servomoteurs recommandés pour ce projet, la ligne de commande est blanche, la ligne d'alimentation positive est rouge, et la ligne de retour est noire. Les lignes du servomoteur doivent être reliées à l'ordinateur de commande et à la source d'alimentation au moyen d'une carte de montage de 20 bandes par neuf trous, laquelle est montée à l'arrière du boîtier de base du bras robot. Coupez la carte de montage à la dimension requise et percez deux trous de montage aux endroits indiqués. Utilisez la carte de montage comme gabarit pour marquer les positions des trous à l'arrière de la base et percez-les. Dans la carte de montage et dans la base, les trous doivent être suffisamment grands pour permettre d'introduire une vis mécanique. Percez un autre trou à l'arrière de la base et insérez le groupe de fils provenant du servo.

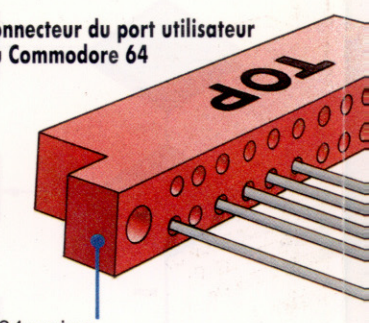
Rassemblez les quatre groupes de fils provenant des servomoteurs à l'arrière de la base et soudez les fils. Les quatre lignes de données et la prise

Étape 1 : connexions de la carte de montage



Étape 2 : connexions a





de terre de l'ordinateur doivent être soudées au bas de la carte de montage. Le Spectrum et le Commodore 64 utilisent cinq fils distincts ou un bout de câble-ruban à cinq voies. Notez que les lignes de commande blanches provenant des servomoteurs sont soudées en position directement au-dessus de chaque ligne de données provenant de l'ordinateur, et que les lignes noires sont toutes soudées sur une seule bande de cuivre située au-dessus des connexions de mise à la terre de l'ordinateur et des deux connexions d'alimentation.

La dernière tâche de soudure consiste à fixer à la carte une source d'alimentation de 5 V. L'alimentation de 5 V fournie par le Commodore 64 ou par le Spectrum n'est pas suffisante pour les quatre servomoteurs dans le cas d'une charge importante. Une autre source doit donc être prévue. Une pile de 4,5 V (ou une batterie de trois piles) est une source d'alimentation idéale. Si on dispose d'un transformateur CC de 5 V, ce dispositif peut également être utilisé. Si vous envisagez d'utiliser des piles, une pince pour accumulateur doit être soudée aux extrémités libres des lignes d'alimentation provenant de la carte de montage. Si vous décidez d'utiliser un transformateur, un connecteur d'alimentation adéquat doit être installé. La borne négative de l'alimentation partage la même bande de cuivre que la prise de terre de l'ordinateur et que les lignes noires de retour des servomoteurs ; la ligne positive d'alimentation est connectée à chacun des fils rouges provenant des servomoteurs via une bande de cuivre commune.

Étape 2 : connexions au port de l'ordinateur

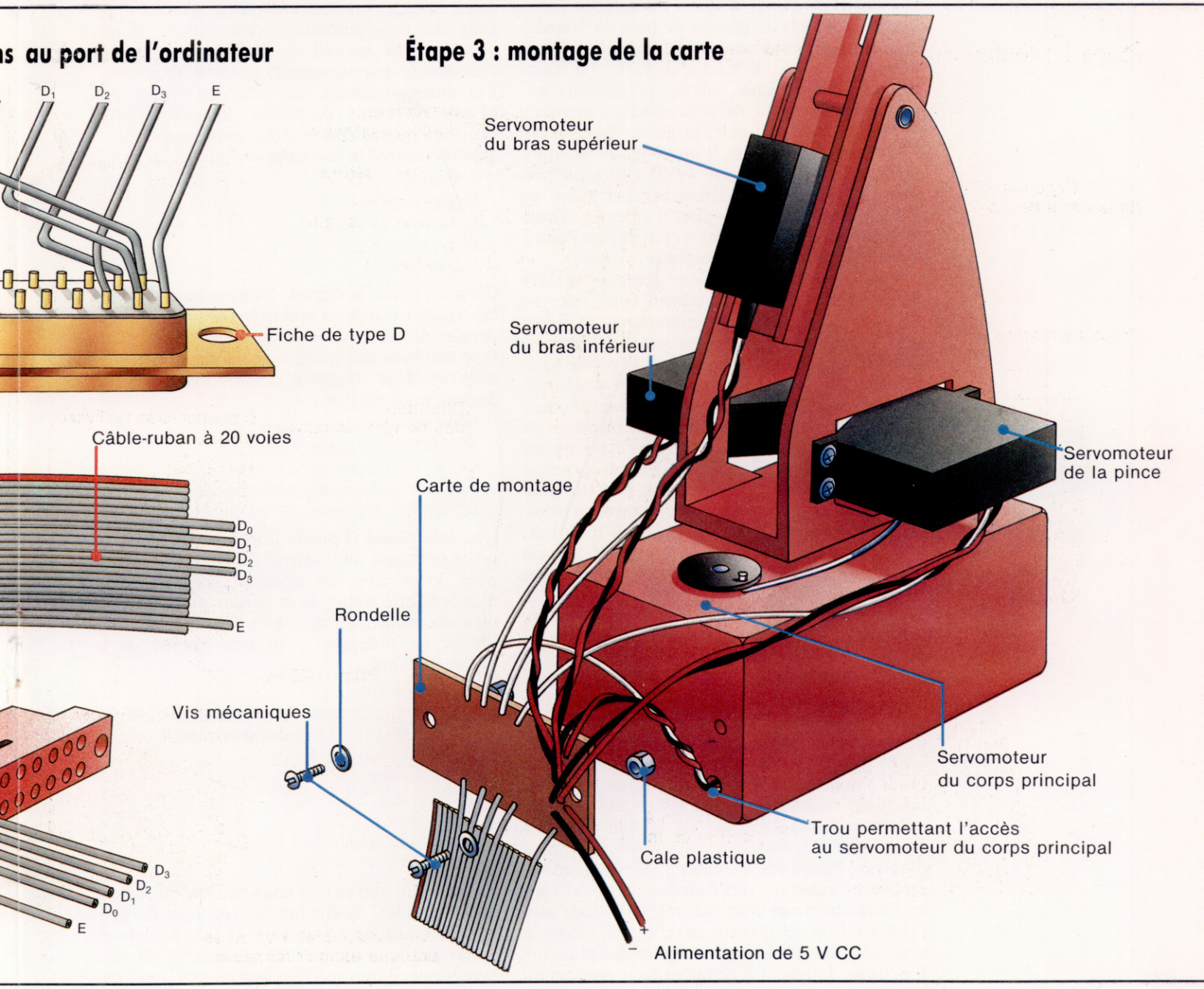
Après avoir terminé le câblage de la carte de montage, le connecteur adéquat de port d'ordinateur doit être fixé aux extrémités libres des lignes de données D_0 à D_3 et à la ligne de prise de terre. Le Commodore 64 utilise un connecteur plat à 24 voies de 0,15 pouce. Puisque ce connecteur se branche dans le port utilisateur dans n'importe quel sens, inscrivez sur l'un des côtés l'indication « haut » avant de commencer. Les cinq lignes provenant de la carte de montage doivent être soudées comme le montre l'illustration.

Le Commodore est doté de circuits intégrés lui permettant de commander des applications par l'intermédiaire d'un port utilisateur. Cependant, le Spectrum ne possède pas de tels circuits, et une interface spéciale doit être construite et branchée dans son port d'extension. Nous avons déjà conçu et construit une telle interface. Les propriétaires de Spectrum doivent donc se référer à ces articles pour construire l'interface.

Les lignes de données et d'alimentation provenant de l'interface sont acheminées au moyen d'une prise de type D à 12 voies, qui se connecte directement dans le robot. Nous pouvons adapter ce connecteur afin de l'utiliser avec le bras-robot. Montez une fiche à 15 voies de type D en utilisant les extrémités libres des lignes de données et de prise de terre provenant de la carte de montage du bras et fixez un couvercle de fiche. Cet adaptateur permet de brancher le bras-robot au port d'extension par l'intermédiaire de la carte d'interface.

Étape 3 : montage de la carte

Lorsque tout le câblage est terminé et soigneusement vérifié, la carte de montage peut être installée à l'arrière de la base en utilisant deux vis mécaniques fixées par des écrous à l'intérieur de la base. Vérifiez que les moteurs sont connectés à la carte de montage dans le bon ordre. La ligne de données D_0 (à l'extrême gauche) doit être connectée à la ligne blanche du servomoteur du corps principal, monté sur la base ; D_1 doit être connecté au servomoteur du bras inférieur, monté du côté gauche du corps principal ; D_2 commande le servomoteur du bras supérieur, monté près du joint de l'épaule et D_3 commande le servomoteur de la pince, monté à la droite du corps principal.



Kevin Jones

Ordre et méthode

En étudiant dBase II d'Ashton-Tate, nous allons voir à quel point ces programmes peuvent se révéler efficaces et puissants.

Pour bien comprendre l'emploi des procédures de recherche, il faut d'abord analyser la façon dont les logiciels de gestion de base de données répondent aux commandes de l'utilisateur. Généralement, SEARCH, LOCATE, DISPLAY, NEXT, LAST (chercher, localiser, afficher, suivant, précédent) constituent des mots clés que le programme prend en compte pour connaître les exigences de celui qui en fait usage. Cela fait, il ouvre le fichier, qu'il parcourt séquentiellement (c'est-à-dire du premier au dernier enregistrement), extrayant au passage les informations dont il a besoin. Toute commande a donc un effet direct, et l'on ne sort pas du format question/réponse.

Mais d'autres logiciels de ce genre — Archive de Psion ou dBase II d'Ashton-Tate, pour ne citer que deux exemples — permettent la rédaction de procédures, ou de programmes entiers, destinés à faciliter la recherche et la collecte des données. Il ne s'agit pas simplement de combiner entre elles des commandes déjà existantes; on est en présence à chaque fois d'un véritable langage de programmation (même s'il reste assez limité), qui fait partie intégrante du programme lui-même. Certains de ces langages sont très semblables à ceux qu'on utilise couramment en informatique, et sont donc très faciles à maîtriser.

Celui de dBase II, s'il reste très spécifique, est pourtant assez proche d'un BASIC structuré (avec quelques mots réservés), pour qu'on puisse l'apprendre et s'en servir avec un minimum de difficultés. La structure d'ensemble en est simple, et comporte des instructions du type DO WHILE, ENDDO, DO CASE...ENDCASE et IF...ENDIF. Le programme comporte par ailleurs une centaine de commandes, dont certaines sont classiques, et d'autres spécialement destinées au traitement des bases de données. Parmi les premières : CALL (pour appeler un programme en langage machine), NOTE (équivalent à REM en BASIC), STORE < expression > to < variable >, qui est en fait la même chose que LET < expression > = < variable >, et bien d'autres. Parmi les secondes : SKIP (pour avancer, vers l'avant ou vers l'arrière) à travers tous les enregistrements d'un fichier), PACK (pour supprimer tel ou tel d'entre eux), et FIND < chaînes de caractères >. dBase II comporte également plusieurs fonctions. Là encore, certaines sont reprises du BASIC : ainsi CHR < expression > (semblable à CHR$(X)) et LEN < chaîne de caractères >. Il y a aussi TYPE < expression > (qui permet de savoir si elle est de type numérique ou alphabétique) et DATE() : cette dernière est une variable système qui conserve la date.

Pour voir à quel point une procédure spécifique se révèle plus maniable, pour extraire une information, qu'une série de commandes entrées directement au clavier, nous allons créer une base de données très simple (un inventaire de pièces mécaniques), puis nous rédigerons, grâce au langage de programmation dont dBase II est pourvu, une procédure qui nous permettra d'obtenir les renseignements que nous désirerons. Les enregistrements du fichier n'auront pas besoin d'être très complexes : ils ne comporteront que quatre zones, trois numériques et une alphabétique. On aura par exemple :

NUMÉRO FABRICANT	06116
NUMÉRO INVENTAIRE	3995
EXEMPLAIRES EN STOCK	86
PRIX	34,75
DESCRIPTION	Bouchon de réservoir

C'est un point de départ. Nous aurons d'abord besoin de trouver un seul mot par zone définie, décider de la longueur de ces zones, puis de leur type (alphanumériques, numériques ou « logiques »). Une structure comme celle qui suit devrait nous suffire :

```
FABRICANT   : 5 caractères; numérique
INVENTAIRE  : 5 caractères; numérique
STOCK       : 3 caractères; numérique
PRIX        : 6 caractères; numérique
DESCRIPTION : 40 caractères; alphanumérique
```

Une fois dBase II lancé, il affiche à l'écran un point signalant qu'il attend l'entrée de vos données. Pour créer un nouveau fichier, il faut d'abord faire usage de la commande CREATE, et bien sûr donner un nom à ce fichier (ce sera ici PIÈCES). Le processus peut être résumé de la façon suivante :

```
• create pieces (CR)
ENTREZ LA STRUCTURE DU CHAMP COMME SUIT
  CHAMP  NOM, TYPE, LARGEUR, CHIFFRES APRÈS LA VIRGULE
001      fabricant, n, 5, 0
002      inventaire, n, 5, 0
003      stock, n, 3, 0
004      prix, n, 6, 2
005      description, c, 40
006      < CR >
```

A noter que tout ce que nous entrons est affiché en minuscules, tandis que les réponses du programme sont en majuscules. Il faut également remarquer que lorsque les zones sont de type numérique, il faut spécifier leur largeur, ainsi que

le nombre de chiffres significatifs après la virgule. Nous n'avons besoin que de cinq zones, et, par conséquent, au lieu d'en définir une sixième, nous nous bornerons à appuyer sur RETURN, ce qui met fin à la phase initiale de création de fichier.

Il faut ensuite, bien sûr, entrer des données, ce que nous ferons à l'aide de la commande APPEND. L'écran est alors effacé, et le « squelette » d'un enregistrement fait son apparition :

```
FABRICANT    :          :
INVENTAIRE   :          :
STOCK        :     :
PRIX         :          :
DESCRIPTION  :
```

Les données sont à ce moment entrées au clavier, et sont affectées successivement à chacune des zones; on passe à la suivante en faisant RETURN après avoir tapé les informations nécessaires. Le programme accepte vos données et attend que vous lui en fournissiez d'autres. Pour marquer la fin d'un enregistrement, on appuie sur RETURN seul au début d'un « squelette » encore vide.

Vous pourrez ensuite exploiter le fichier que vous avez créé. Supposons par exemple que vous désiriez connaître quel est le montant total des prix de vente de toutes les pièces composant votre stock. Un moyen d'y arriver consisterait à examiner successivement chaque enregistrement et à relever, « à la main », le nombre d'exemplaires en stock de chacune des pièces, le prix correspondant, puis à multiplier le premier par le second. Après quoi, il faudrait faire le total de l'ensemble en additionnant toutes les sommes ainsi obtenues sur l'ensemble du fichier. C'est d'ailleurs ainsi que, pendant très longtemps, il a fallu procéder.

Avec les logiciels de gestion de bases de données, heureusement, les choses sont plus simples. dBase II réduit toute l'opération à un programme de deux lignes :

```
. use pièces
. sum inventaire * prix
37870,58
```

USE est une commande qui enjoint au programme de travailler sur un fichier nommé PIÈCES. Sum revient à lui dire : « Additionner, pour l'ensemble des enregistrements, le résultat de la multiplication de STOCK par PRIX. » Le résultat est alors affiché. Les opérations de calcul sont très rapides (et sans erreurs, dès lors que les données sont correctes). On voit que dBase II permet un gain de temps énorme et simplifie considérablement des recherches fastidieuses au cours desquelles il est à peu près inévitable de se tromper.

Encore n'avons-nous fait usage que des ressources « de base » du programme. Il est possible d'aller encore plus loin en intégrant plusieurs procédures au sein d'un même programme. Voici ce qu'il faudrait faire, de ce point de vue, pour en créer un qui s'appellerait VALEUR :

```
. Modify commande
ENTREZ NOM DE FICHIER : valeur
set talk off
use pièces
sum stock * prix
set talk on
cancel
```

Le tout sera ensuite stocké sur disquette, et pourra être appelé à tout moment à partir de dBase II, en tapant simplement :

```
. do valeur
```

Il sera alors chargé en mémoire et exécuté, puis, dès qu'il sera arrivé à son terme, le contrôle repassera automatiquement à dBase II.

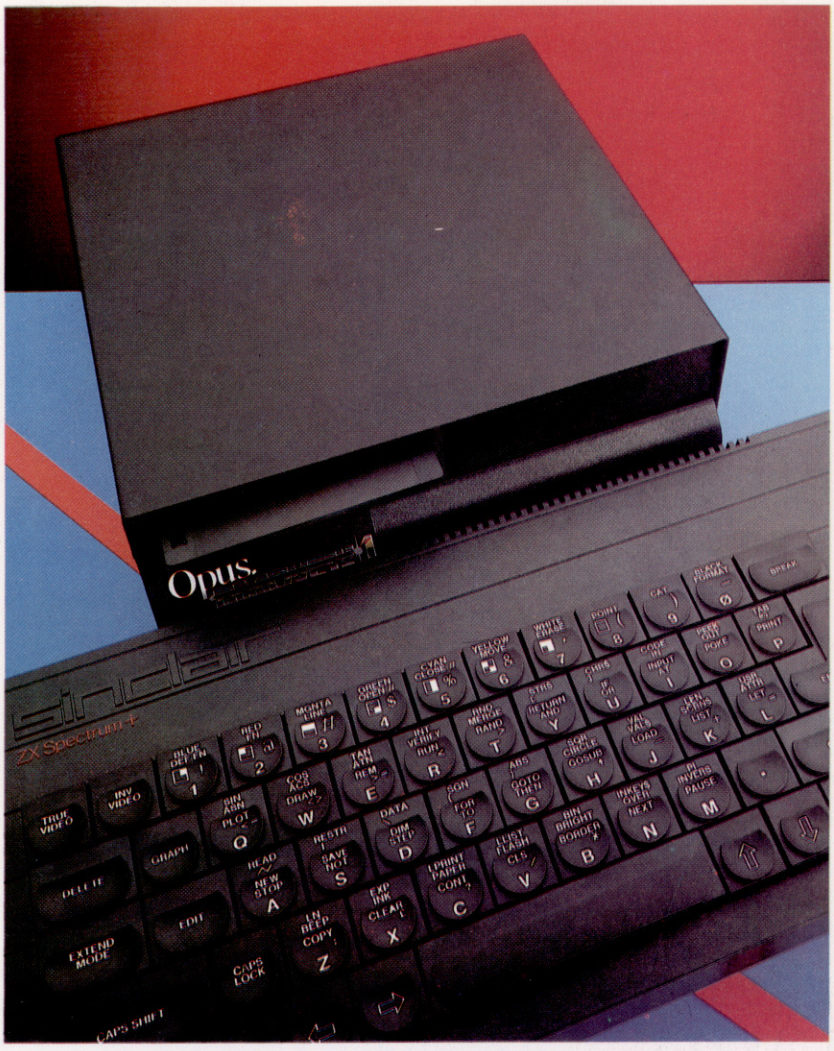
Quel est ton nom?

```
* Affichage noms
SET TALK OFF
USE MEMBRES
DO WHILE. NOT. EOF
  IF NOM=«DUMAS»
    DISPLAY NOM
  ENDIF
  SKIP
ENDDO
```

Le programme ci-dessus est très simple; il est extrait d'un fichier consacré aux membres d'une association de tous ceux dont le nom est Dumas. Il est rédigé dans le langage de programmation dont dBase II est pourvu.
La première ligne commence par un astérisque; cela indique qu'il s'agit d'une ligne de commentaire. Nous n'avons pas besoin de faire apparaître à l'écran tous les numéros des enregistrements, pendant que dBase II les parcourt successivement; c'est pourquoi nous préciserons SET TALK OFF.
La base de données peut par ailleurs comporter plusieurs fichiers, et c'est pourquoi nous devrons spécifier celui qui nous intéresse, ici, MEMBRES. Comme il doit être exploré dans son intégralité, nous mettrons au point une boucle de type DO WHILE...ENDDO. EOF signifie « fin du fichier » (End Of File). A l'intérieur de cette boucle, nous établirons une condition : chaque fois que le programme rencontre la chaîne de caractères DUMAS dans le champ attribué au nom, il affiche le résultat. IF doit, bien entendu, être suivi de ENDIF. dBase II passe alors à l'exécution de la commande SKIP, qui le fait passer à l'enregistrement suivant.

Discovery 1 de Opus

Nous terminons notre étude sur les alternatives au Microdrive de Sinclair avec le Discovery 1 d'Opus, après avoir vu le Wafadrive de Rotronics.



Nouvelles méthodes
Le Discovery est un système d'extensions complet destiné à l'utilisateur du Spectrum. La machine comprend un lecteur de disques, un système d'exploitation de disques (SED), et des connecteurs pour les liaisons imprimante, manettes de jeu, moniteur mixte, et autres périphériques, sans interfaces.

Si le Wafadrive s'est révélé plus fiable que le Microdrive, il est également apparu plus lent. En outre, son principe repose toujours sur le système, douteux, de la bande en continu. Nous étudions ici une approche plus conventionnelle de la mémoire de masse : un système utilisant des disques. Il est produit par Opus Supplies.

Un lecteur de disque pour le Spectrum n'est pas en soi une idée nouvelle. Il y a eu ces dernières années un certain nombre de systèmes d'exploitation de disquettes (Disk Operating System/SED) et d'interfaces destinés à cette machine. Aucun d'entre eux ne s'est imposé. Cela est partiellement dû au fait que ces interfaces ont été seulement annoncées dans la presse spécialisée, ce qui leur a donné une connotation très particulière et a semblé indiquer qu'ils étaient uniquement destinés aux inconditionnels du Spectrum et aux programmeurs en code machine. La deuxième raison de leur insuccès fut leur diffusion exclusive par commandes postales. Ils ont donc manqué de cet appui commercial nécessaire pour imposer un produit dans l'esprit du public.

Le Discovery 1 est en métal. Il se place à l'arrière du clavier qu'il domine légèrement pour faciliter l'introduction de disquettes dans son lecteur. La liaison avec le Spectrum se fait par une case à cartouche via le port d'extension. Bien qu'il paraisse facile à réaliser, le branchement du Discovery au connecteur latéral du Spectrum n'est pas évident. Cela est dû au câble et à la cassette elle-même qui gênent. Ce problème semble anodin, mais il faut savoir qu'un connecteur latéral mal branché peut endommager sérieusement le Spectrum.

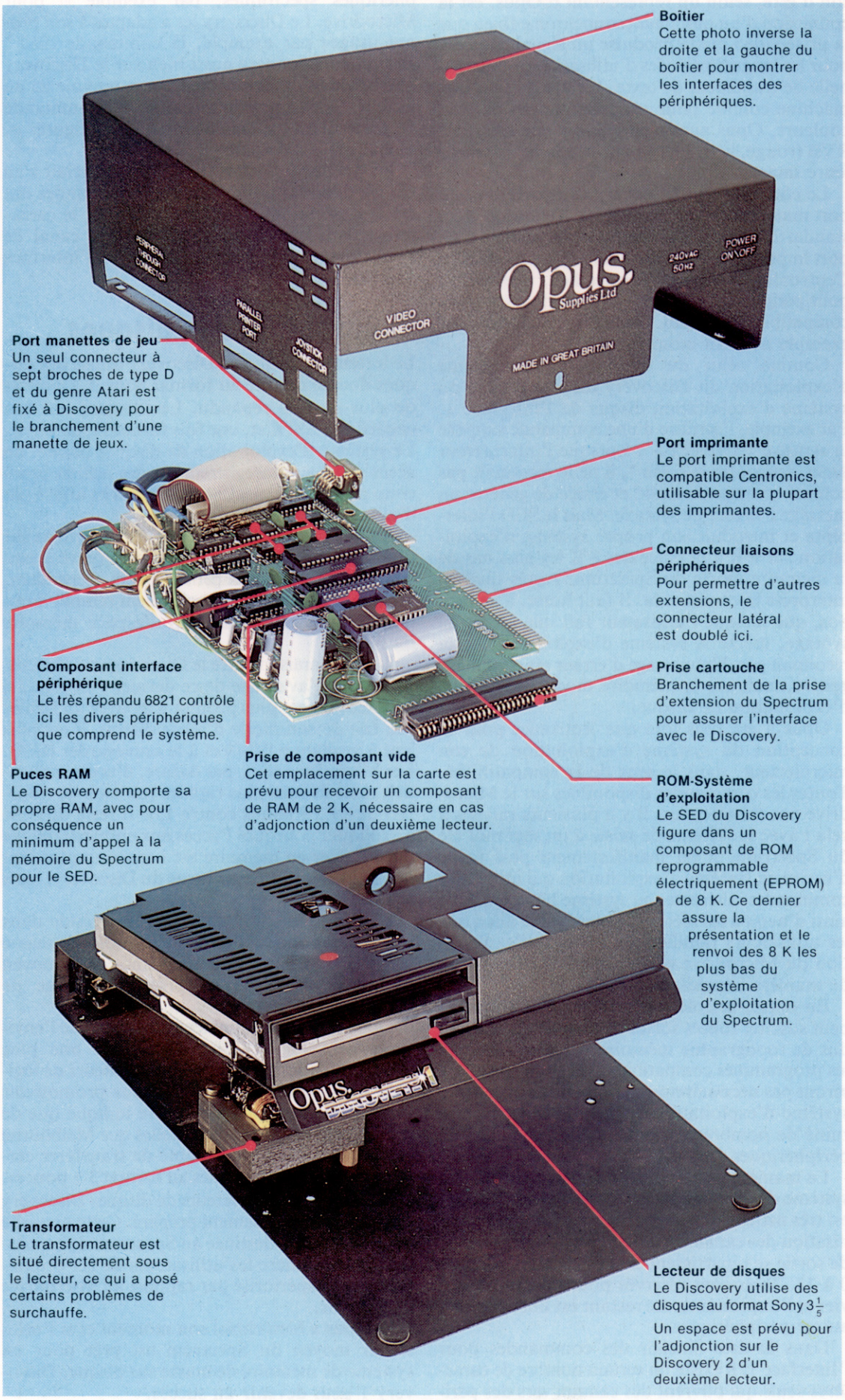
La difficulté est du reste moindre avec le Spectrum proprement dit qu'avec le Spectrum +, plus volumineux. On peut d'ailleurs prendre plusieurs minutes avant d'être sûr que le périphérique est branché de manière appropriée. Une fois la liaison réalisée, le Spectrum passe sous l'alimentation du Discovery. Ce dernier a en effet été conçu pour s'alimenter lui-même, mais aussi le Spectrum, rendant superflue l'alimentation de la machine.

Extension double lecteur

Au-dessus de la case de la cartouche, à gauche, est situé le lecteur simple de 3 $\frac{1}{2}$ pouces. A sa droite, un espace a été ménagé pour recevoir un deuxième lecteur. Opus compte lancer une version double lecteur appelée Discovery 2. Les utilisateurs de Discovery 1, désireux d'étendre leur système à un double lecteur, attendront la sortie d'une configuration externe du lecteur de disque devant être appelée Discovery +.

Ces utilisateurs ne devraient pas pour autant être cantonnés aux lecteurs Opus puisque les fabricants de Discovery affirment que les disques 5 $\frac{1}{4}$ standard peuvent être également mis en œuvre de la même manière.

L'esprit dans lequel Discovery a été conçu est à rapprocher de celui qui caractérise Centronics : fournir non seulement une mémoire de masse, mais aussi des interfaces pour des périphériques supplémentaires permettant aux utilisateurs d'exploiter des imprimantes et d'autres systèmes. Une case de connexion moniteur vidéo mixte figure à l'arrière du Discovery 1. Sa rai-



Boîtier
Cette photo inverse la droite et la gauche du boîtier pour montrer les interfaces des périphériques.

Port manettes de jeu
Un seul connecteur à sept broches de type D et du genre Atari est fixé à Discovery pour le branchement d'une manette de jeux.

Port imprimante
Le port imprimante est compatible Centronics, utilisable sur la plupart des imprimantes.

Connecteur liaisons périphériques
Pour permettre d'autres extensions, le connecteur latéral est doublé ici.

Composant interface périphérique
Le très répandu 6821 contrôle ici les divers périphériques que comprend le système.

Prise cartouche
Branchement de la prise d'extension du Spectrum pour assurer l'interface avec le Discovery.

Puces RAM
Le Discovery comporte sa propre RAM, avec pour conséquence un minimum d'appel à la mémoire du Spectrum pour le SED.

Prise de composant vide
Cet emplacement sur la carte est prévu pour recevoir un composant de RAM de 2 K, nécessaire en cas d'adjonction d'un deuxième lecteur.

ROM-Système d'exploitation
Le SED du Discovery figure dans un composant de ROM reprogrammable électriquement (EPROM) de 8 K. Ce dernier assure la présentation et le renvoi des 8 K les plus bas du système d'exploitation du Spectrum.

Transformateur
Le transformateur est situé directement sous le lecteur, ce qui a posé certains problèmes de surchauffe.

Lecteur de disques
Le Discovery utilise des disques au format Sony $3\frac{1}{5}$

Un espace est prévu pour l'adjonction sur le Discovery 2 d'un deuxième lecteur.

DISCOVERY 1

PRIX

Discovery 1 : ***
Discovery + extension : **
Discovery 2 : ****

DIMENSIONS

300 × 210 × 75 mm.

INTERFACES

Connecteur liaisons périphériques, interface parallèle Centronics, port manette de jeux, prise vidéo mixte.

FORMAT

Disques standard Sony $3\frac{1}{2}$ pouces, simple face, double densité.

CAPACITÉ

250 K au total, 180 K une fois formaté.

VITESSE

Vitesse de transfert 15 Kbauds, accès piste à piste : 3 millisecondes.

son d'être, selon un représentant d'Opus, est la connexion d'un moniteur monochrome (bien que la prise vidéo mixte produise un signal couleur) pour les longues périodes d'utilisation professionnelle de traitement de texte. Il reste que sur une machine connue pour ses programmes de jeux couleurs, Opus aurait pu prévoir une interface RVB (rouge, vert, bleu) pour produire une meilleure image.

Le côté droit du Discovery comporte un seul port manettes de jeu, compatible Kempston et au standard Atari ; à côté de ce dernier se trouve un port imprimante parallèle bidirectionnelle de type Centronics. Il y a, en dernier lieu, une liaison pour périphérique destinée aux interfaces avec les compatibles Spectrum. Cela permet la liaison par exemple avec un moniteur RVB.

Comme celui du Wafadrive, le système d'exploitation du Discovery est très proche du système d'exploitation disque de l'Interface 1. Par exemple, l'écriture d'une commande suppose la syntaxe <COMMANDE>*. Lorsque l'interpréteur BASIC rencontre le signe *, il ne le reconnaît pas comme commande BASIC et essaie de générer un message d'erreur de syntaxe. Mais le SED l'intercepte et introduit son propre système d'exploitation sur 8 K à la place des 8 K les plus bas de la mémoire morte du Spectrum. Après quoi, il interprète la commande. Il faut noter, à ce propos, que lorsque l'utilisateur fait une erreur de syntaxe, même le système d'exploitation ne la reconnaît pas. Un message d'erreur est cependant généré, mais via la mémoire morte du système d'exploitation de disque.

Opus a été plus loin que Rotronics pour la conception du système d'exploitation de son microlecteur, dans le sens de la compatibilité. Toutes les commandes disponibles sur le Microdrive ont été retenues. Il y a plusieurs raisons à cela : avec le système de saisie à un seul mot clé du Spectrum, il est manifestement plus facile d'écrire un système d'exploitation qui utilise des commandes inhérentes au système hôte. On évite ainsi d'avoir à les réécrire. Cela signifie aussi que les utilisateurs familiers du système d'exploitation du Microdrive pourront utiliser Discovery de manière immédiate.

En outre, bricoler un système d'exploitation peut susciter toutes sortes de problèmes inattendus de topographie mémoire. Cela signifie que les programmes compatibles avec l'interface 1 ne seront pas nécessairement compatibles avec votre système d'exploitation bricolé, problème qui a ruiné de nombreux fabricants indépendants de périphériques.

La manière dont l'Opus a étroitement suivi le système de commandes du Microdrive de Sinclair est très nettement perceptible au niveau de l'organisation des canaux. Sur le Spectrum, les canaux de sortie se répartissent en 16 flux numérotés de 0 à 15. Trois sont réservés pour l'écran, le clavier et l'imprimante. Le restant est libre pour les autres périphériques.

Dans la liste Sinclair des commandes pour l'Interface 1, il existe un certain nombre de caractères seuls qui ouvrent des canaux vers des périphériques spécifiques, par exemple m pour Microdrive. Le Discovery les a adaptés à son propre usage ; par exemple, la commande LOAD * 'm';1'nom' fonctionnera aussi bien sur le Discovery que sur le Microdrive, bien que m renvoie ici au lecteur. Opus a néanmoins adapté la commande de sorte que l'on puisse ne pas faire figurer m, ce qui offre davantage de facilité.

Le système un peu compliqué du Sinclair s'en trouve donc simplifié. D'autres commandes ont été également adaptées : par exemple, le caractère t dans une commande ouvre le canal de l'imprimante au lieu de celui de l'interface RS232.

Exploitation des disques

Le lecteur de disque de Discovery utilise des disques double densité au format Sony 3 ½ pouces de plus en plus répandu. Les disques ont une capacité de 250 K et, une fois formatés, de 180 K. Le système d'exploitation de disques permet un accès direct aux enregistrements des fichiers beaucoup plus rapide que l'accès séquentiel utilisé par d'autres systèmes de disques.

Il n'y a pas de limite au nombre de fichiers sur disque, ce qui est important lorsque l'on veut sauvegarder de nombreux petits fichiers. Cependant, lorsque le répertoire est rapidement rempli, il restera certainement beaucoup d'espace mémoire inutilisé.

La comparaison entre le Discovery et le Microdrive est à l'avantage de ce dernier : en effet, si le Discovery est plus prompt à trouver un fichier (du fait de son mode d'accès direct), il est plus lent à le charger (LOAD) et à le sauvegarder (SAVE). Les Microdrive sont, par nature, d'accès séquentiel, mais ils sont plus rapides (un taux de transfert de 19,2 Kbauds contre 15). Il reste difficile à expliquer pourquoi l'accès aux enregistrements d'un fichier en mémoire est beaucoup plus lent que sa localisation dans le cas du Discovery, mais le fait est là.

L'intérêt principal de ce système réside dans son système de sauvegarde de mémoire de masse plus fiable, et dans la nature même de son moyen de sauvegarde plus universel (davantage de fournisseurs).

Opus s'est manifestement préoccupé de l'environnement logiciel du Discovery : une plus grande diffusion signifie pour les éditeurs de logiciels une vitrine intéressante pour promouvoir leurs produits. Opus a en outre indiqué que de grosses sociétés de logiciels, telles que Melbourne House et Legend ont accepté de transférer certains de leurs programmes au format 3 ½ pouces.

Le lancement des lecteurs de disques Discovery a donc été soigneusement préparé. Cette nouvelle ligne de lecteurs destinée au Spectrum doit maintenant convaincre les utilisateurs de cet ordinateur de sa supériorité par rapport au Microdrive de Sinclair.

Si Opus a bien choisi son moment et si l'utilisateur moyen du Spectrum est prêt pour un système de mémoire de masse sur disque, Discovery 1 peut devenir un succès.

Ports d'appel

Voyons en détail les ports périphériques du Commodore 64, une brève description de l'accès aux routines E/S du noyau et montrons comment utiliser les routines du SE pour commander le RS232.

Notre photo montre tous les ports disponibles à l'arrière du Commodore 64. Comme le port cassette, le port audio/vidéo et la prise TV, il y a trois ports E/S disponibles pour le programmeur. Ce sont, de gauche à droite, le port d'extension (également connu sous le nom de port cartouche), le port série (ou bus série) et le port utilisateur.

• *Le port série :* les imprimantes série Commodore et le lecteur de disque série 1541 se branchent sur cette prise DIN à six pointes. L'instruction OPEN s'applique toujours à ce port avec n'importe quel chiffre différent de 2 (le 2 se rapportant à RS232 via le port utilisateur). Par exemple, avec le numéro 8, la commande OPEN 2,8,2, «NOMFICHIER» ouvrira un fichier au lecteur de disque à travers le port série, sauf avec les appareils Commodore, par BASIC ou les routines de noyau.

Ce sont des interfaces disponibles, cependant, qui acceptent des messages série via ce port et effectuent une conversion série/parallèle IEEE. Cela permet d'utiliser sur le C64 des périphériques qui s'interfacent avec le PET de Commodore. De tels périphériques incluent le lecteur de disque 4040.

• *Le port utilisateur :* c'est un connecteur plat à vingt-quatre voies, 0,15 pouces, qui peut servir à la fois pour les communications parallèle et série. Par exemple, ce port peut être utilisé pour connecter le C64 à une imprimante d'un autre fabricant (notamment Epson), qui est traitée comme un dispositif RS232.

Le port utilisateur peut également servir à la communication 8 bits parallèle, mais il faut alors écrire — ou obtenir — son propre logiciel de commande, puisque le SE n'a pas de routines de commande (programmes en langage machine qui contrôlent les périphériques) pour la communication parallèle. Prochainement, nous fournirons un programme appelé « Parawedge » (para-coin), qui illustre l'ensemble des tâches requises pour obtenir un programme de communication parallèle commandé par interruption. Celui-ci peut servir à transférer un bloc de mémoire de l'ordinateur à un dispositif externe (tel qu'un programme BASIC d'un C64 à un autre), pourvu que les deux machines aient des niveaux 5 v.

• *Le port d'expansion :* le port d'expansion est un connecteur femelle à 44 broches, qui donne accès à tout le contrôle principal, lignes d'adresses et de données sur le C64. Ce port est employé pour interfacer les cartouches de jeux et les cartouches IEEE parallèle, qui permettent de connecter l'ordinateur aux périphériques PET. Il sert aussi pour tout autre dispositif ou logiciel qui requiert un contrôle presque complet de la machine, ou des programmes suffisamment bons pour garantir la conception et la construction d'une carte pour les contenir sous forme de ROM ou d'EPROM.

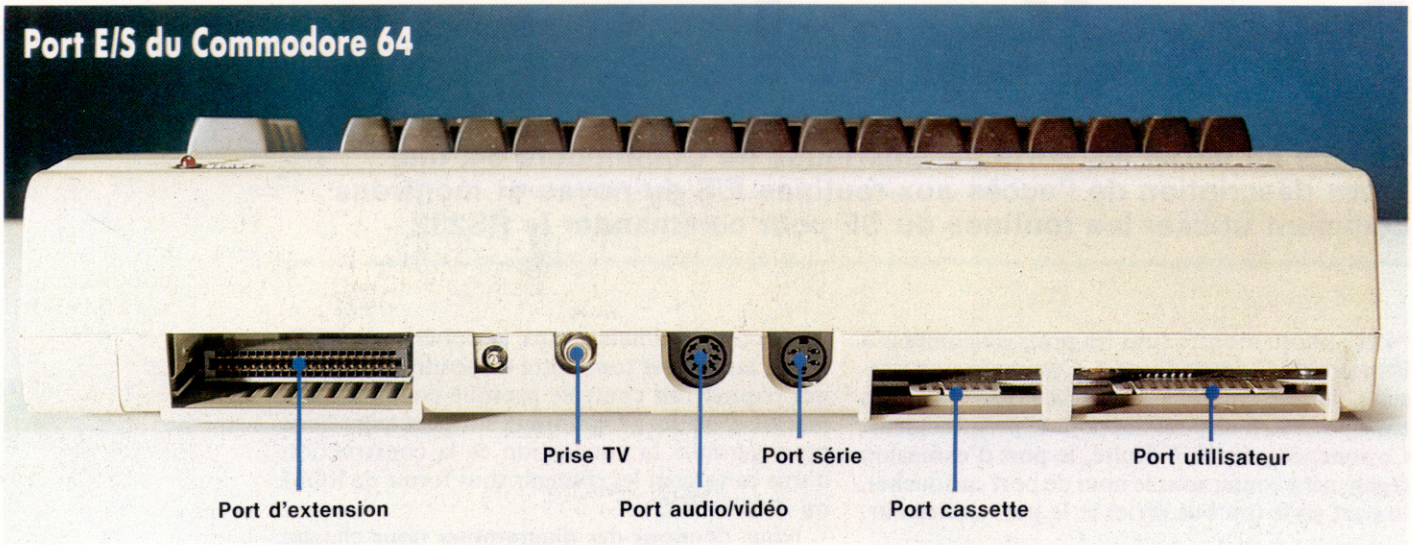
Nous donnons des diagrammes pour chacun de ces trois ports, montrant les fonctions de leurs connexions. Considérons maintenant des aspects de l'utilisation de ces ports par le SE. L'ensemble des routines SE qui manient les E/S est appelé le noyau *(kernel)*. Ce sont des programmes en langage machine appelés par les instructions BASIC d'entrée/sortie OPEN, CLOSE, GET#, PRINT#, etc. Commodore s'est arrangé pour que les routines de noyau soient d'accès facile pour le programmeur système en langage machine. Nous donnons un court listage en langage machine qui nous permet d'utiliser la routine LOAD de noyau.

Nous examinons aussi les possibilités du RS232 incorporé dans le C64, et nous considérons comment il peut être employé, par exemple, pour commander un modem via les routines RS232.

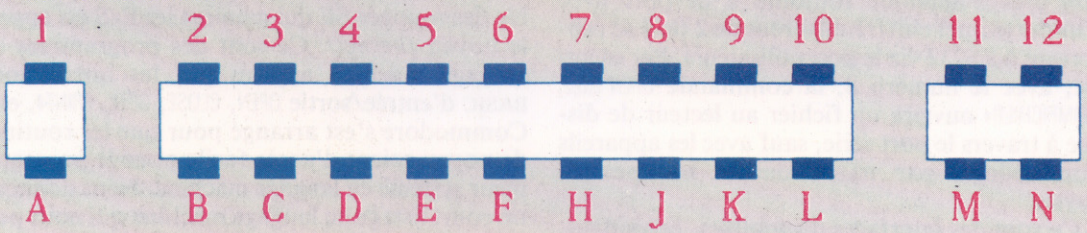
Il ne sera pourtant pas toujours possible d'utiliser les routines système pour les E/S. Parfois, vous pourrez avoir besoin d'utiliser la communication parallèle et des débits de données élevés vers un de vos périphériques, plutôt qu'une communication série vers un dispositif plus éloigné. Il n'existe pas de routine système pour commander le port utilisateur sur le C64. Ainsi, pour réaliser une communication parallèle, il faut programmer vous-même les deux adaptateurs d'interface complexe 6526 (CIA). Nous vous aiderons, dans le prochain numéro, à commencer une routine en assembleur pour un transfert de données parallèle 8 bits. Cette routine permet à l'ordinateur de lire ou d'envoyer des données via le port utilisateur, tout en traitant simultanément un programme BASIC. La magie du temps partagé s'accomplit en faisant en sorte que le code qui lit ou écrit les données est commandé par interruption.

Nous avons déjà donné dans cette série d'articles un exemple de « coin » de demande d'interruption IRQ. Dans le cas présent, la routine est commandée par NMI parce que la ligne de drapeau de port utilisateur peut servir à générer une interruption NMI (non masquable) : c'est la seconde ligne d'interruption pour le processeur 6510. Au contraire de IRQ, un signal d'interruption sur la ligne NMI ne peut pas être masqué à l'aide des commandes SEI et CLI.

Port E/S du Commodore 64



Port utilisateur



Haut

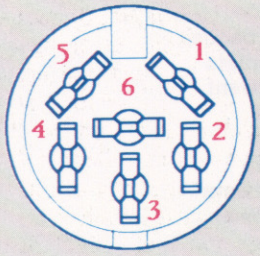
Broche	Type	Fonction générale
1	GND	Terre
2	—	+ 5 V à 100 mA maximum
3	—	Réinitialisation système
4	CNT1	Compteur port série CIA#1
5	SP1	Port série CIA#1
6	CNT2	Compteur port série CIA#2
7	SP2	Port série CIA#2
8	PC2	Ligne d'entrée en communication de CIA#2
9	—	Cette ligne est connectée à ATN sur le port série
10	—	9 V ac + phase
11	—	9 V – phase
12	—	Terre

Bas

Broche	Type	Fonction RS232 (*)	Utilisé en ligne 3/ligne X
A	GND	Terre protection	Les deux
B	FLAG2	Donnée reçue — entrée	Les deux
C	PB0	Donnée reçue — entrée	Les deux
D	PB1	Demande d'envoi (RTS) — sortie	Les deux (ht dans 3-1)
E	PB2	Fin de données (DTR) — sortie	Les deux (ht dans 3-1)
F	PB3	Indicateur d'anneau (RI) — entrée	—
H	PB4	Signal ligne reçue (DCD) — entrée	Ligne X seulement
J	PB5	Non attribué	—
K	PB6	Prêt à envoyer (CTS) — entrée	Ligne X seulement
L	PB7	Données prêtes (DSR) — entrée	Ligne X seulement
M	PA2	Données transmises — Sortie	Les deux
N	GND	Terre signal	Les deux

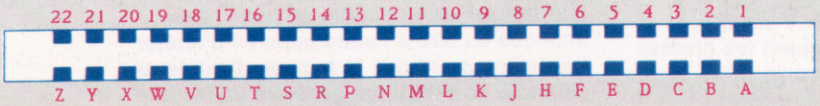
Notes :
1. Ligne 3 = protocole Xon/Xoff.
Ligne X = protocole CTS/RTS.
2. (*) En plus d'être adressé via les routines noyau SE RS232, ces lignes peuvent aussi être programmées directement pour les E/S définies par l'utilisateur. Ces connexions étaient utilisées pour contrôler les dispositifs externes comme le tampon et le robot de sol dans notre série « Boîte à outils ».

Port série



Broche	Type	Commentaires
1	SERIAL SRQIN	Service demandé
2	GND	
3	SERIAL ATN 1/0	Signal d'attention aux appareils sur port
4	SERIAL CLK 1/0	Horloge de synchronisation pour port série
5	SERIAL DATA 1/0	Ligne de transfert données 1 bit
6	RESET	Ligne réinitialisation matériel

Port d'extension



Haut

Broche	Type	Commentaires
1	GND	
2	+ 5 V	
3	+ 5 V	
4	IRQ	Demande d'interruption
5	R/W	Validation lecture/écriture
6	Dot clock	
7	I/O 1	
8	GAME	
9	EXROM	
10	I/O 2	
11	ROML	
12	BA	
13	DMA	
14	D7	
15	D6	
16	D5	
17	D4	8 lignes données
18	D3	
19	D2	
20	D1	
21	D0	
22	GND	

Bas

Broche	Type	Commentaires
A	GND	
B	ROMH	
C	RESET	Ligne réinitialisation matériel
D	NMI	Interruption non masquable
E	S02	
F	A15	
H	A14	
J	A13	
K	A12	
L	A11	
M	A10	
N	A9	
P	A8	16 lignes adresses
R	A7	
S	A6	
T	A5	
U	A4	
V	A3	
W	A2	
X	A1	
Y	A0	
Z	GND	

Le noyau en action

Le noyau *(kernel)* est une bibliothèque de sous-programmes d'E/S, qui sont appelés à partir de BASIC ou par langage machine écrit par l'utilisateur. La ROM noyau est située entre $E000 et $FFFF, mais les routines proprement dites sont appelées à partir d'une « table de saut » — c'est-à-dire une table de pointeurs des emplacements des routines, contenue en haut de la mémoire. L'avantage qu'il y a à appeler les routines depuis la table de saut est que le code peut être plus facilement utilisé avec d'autres machines Commodore, parce que les adresses de la table de saut ne varient pas, bien que les contenus puissent changer.

Pour utiliser les programmes de noyau, il vous faudra de la matière de référence, puisque chaque routine a besoin d'être soigneusement établie avant d'être appelée. Le guide de référence du programmeur Commodore est probablement la meilleure source; les routines de noyau sont documentées aux pages 268 à 306.

Par exemple, pour utiliser la routine de noyau LOAD pour effectuer une réallocation, on charge (LOAD) généralement le fragment de code suivant. Placez les codes ASC (version Commodore de ASCII) du nom de fichier dans des adresses consécutives en mémoire, puis utilisez les parties de code suivantes :

```
LDA   #$01     Numéro de fichier logique (lfn)
LDX   #$08     Numéro d'appareil (disque)
LDY   #$00     Adresse secondaire (#$00 donne
               le chargement de réallocation)
JSR   $FFBA    Routine de noyau SLFS (mettre lfn
               et adresse secondaire)
LDA   #$0A     Longueur de nom de fichier
               (par exemple 10 décimal)
LDX   PLO      Pointeur Octet-lo pour adresse
               de début de nom de fichier
LDY   PHI      Pointeur Octet-hi pour adresse
               de début de nom de fichier
JSR   $FFBD    Routine de noyau SETNAM
               (mettre nom de fichier)
LDA   #$00     Charger =#$00 / Vérifier =#$01
LDX   DLO      Octet-lo adresse de début destination
LDY   DHI      Octet-hi adresse de début destination
JSR   $FFD5    Routine de noyau LOAD
```

Utilisation de RS232

Du moment qu'on adopte une approche méthodique, il n'y a pas de problème sérieux dans l'utilisation des routines SE pour commander RS232, soit à partir de BASIC, soit du langage machine. Il y a différents points à considérer pour se préparer à utiliser RS232 par le port utilisateur. A part le débit en bauds, sur lequel nous reviendrons, le seul autre problème que vous risquez de rencontrer concerne les points suivants :

• Le C64 fonctionne aux niveaux 0 V à 5 V, alors que le standard RS232 requiert les niveaux – 12 V à + 12 V. Donc, à moins de communiquer avec un autre C64, il sera nécessaire de construire ou d'acquérir une possibilité de « conversion de niveau ». Commodore fournit une cartouche RS232 pour accomplir cette fonction.

• Les codes ASC Commodore diffèrent des codes ASCII standard, aussi sera-t-il nécessaire d'établir deux tableaux de conversion : un pour transmettre et un pour recevoir.

• Chaque fois qu'un canal RS232 est ouvert (OPEN), le SE efface (CLEAR) automatiquement les registres. Les valeurs de toutes les variables utilisées précédemment par un programme BASIC sont immédiatement perdues; les commandes GOSUB produisent des messages d'erreur lorsqu'on arrive à RETURN. C'est parce que la routine de noyau RS232 alloue deux tampons à 256 octets en haut de la mémoire. Cela, en soi, peut causer un problème parce que, s'il n'y a pas assez d'espace en haut de la mémoire pour les tampons, votre programme sera altéré et les erreurs ne seront pas reportées.

• Si un programme BASIC est long ou s'il fait beaucoup d'attributions de chaînes, alors, tôt ou tard, il y aura une masse d'erreurs. Les données entrées pourront alors être perdues. Cependant, cela n'arrivera pas souvent si le programme est relativement court et ne fait pas beaucoup d'attributions de chaînes. Pour ouvrir (OPEN) un canal RS232 à partir de BASIC, on utilise le format suivant :

```
OPEN 2,2,3,CHR$(CTRL)+CHR$(COMM)
```

CTRL et COMM sont les octets de contrôle et de commande qui contiennent l'information requise pour mettre le canal. Notez que CTRL et COMM doivent être littéralement deux octets (ou les PEEK de deux emplacements préalablement remplis), et pas deux variables ! Chaque bit dans les octets CTRL et COMM a une fonction, comme indiqué dans les tableaux.

Par exemple, pour ouvrir un canal RS232 avec un bit d'arrêt, une longueur de mot de 7 bits, 300 bauds (total octets CTRL = 38), parité paire, duplex bidirectionnel et contrôle de flux ligne 3 (total octets COMM = 96), nous utilisons la commande :

```
OPEN 2,2,3,CHR$(38)+CHR$(96)
```

Lorsqu'on décide quel débit en bauds utiliser, les facteurs suivants doivent être retenus. Lorsqu'on envoie l'information, le débit en bauds n'est pas critique, parce que d'autres dispositifs admettent généralement des variations de vitesse.

On a envoyé avec succès des caractères de BASIC via RS232 à un débit atteignant 2 400 bauds. Bien sûr, tandis que le débit de bits par octets est de 2 400, l'espace entre octets est souvent plus long, de sorte que le débit effectif de transfert est bien inférieur.

La situation est néanmoins toute différente à la réception de données. Dans ce cas, même à 300 bauds, un programme BASIC aura à peine le temps de recueillir un octet du tampon d'entrée et de l'imprimer sur l'écran dans une boucle simple. Pour recevoir effectivement, il faut utiliser un « contrôle de flux ». Cela implique de dire à l'autre appareil d'arrêter l'émission avant que le tampon d'entrée ne déborde. La procédure générale utilisant le protocole ligne 3 est :

1. Lire un petit nombre d'octets (plus le débit est élevé, plus ce nombre devra être petit) en utilisant GET#2,A$. Il faut les traiter rapidement ou les stocker sur tampon dans un tableau.
2. Arrêter l'envoi par les autres appareils à l'aide de la commande : PRINT#2, CHR$(17).
3. Lire les octets suivants jusqu'à ce que le tampon RS232 soit vide. Traiter tous les octets lus, en vérifiant si des touches sont appuyées ou s'il y a des signaux fin de fichier (EOF).
4. Permettre au dispositif de reprendre l'émission en utilisant PRINT#2, CHR$(19) et revenir à la première étape.

Une fois ouvert (OPEN) un canal RS232, des octets sont envoyés ou reçus normalement en utilisant PRINT# ou GET#. Il faut éviter INPUT#. L'octet d'état, ST, doit être vérifié, comme pour tout programme comprenant des E/S — l'état d'erreur nul est désigné par ST=0 ou ST=8. Enfin, on peut s'attendre à ce que CLOSE produise une autre autoréinitialisation, puisque les tampons sont désaffectés.

Octet CTRL

Bit 7	6	5	4	3	2	1	0	Fonction
—	—		X	0	0	0	1	50 bauds
—	—	—	X	0	0	1	0	75 bauds
—	—	—	X	0	0	1	1	110 bauds
—	—	—	X	0	1	0	0	134,5 bauds
—	—	—	X	0	1	0	1	150 bauds
—	—	—	X	0	1	1	0	300 bauds
—	—	—	X	0	1	1	1	1 200 bauds
—	—	—	X	1	0	0	0	1 800 bauds
—	—	—	X	1	0	1	0	2 400 bauds
—	0	—	X	—	—	—	—	Donnée 8 bits
—	0	0	X	—	—	—	—	Donnée 7 bits
—	1	1	X	—	—	—	—	Donnée 6 bits
—	1	0	X	—	—	—	—	Donnée 5 bits
0	—	1	X	—	—	—	—	1 bit d'arrêt
1	—	—	X	—	—		—	2 bits d'arrêt

X = sans intérêt

Octet COMM

Bit 7	6	5	4	3	2	1	0	Fonction
—	—	—	—	X	X	X	0	Protocole ligne 3
—	—	—	—	X	X	X	1	Protocole ligne X
—	—	—	0	X	X	X	—	Duplex bidirectionnel
—	—	—	1	X	X	X	—	Semi-duplex bidirectionnel
—	—	0	—	X	X	X	—	Parité désactivée
0	0	1	—	X	X	X	—	Parité impaire
0	1	1	—	X	X	X	—	Parité paire
1	0	1	—	X	X	X	—	Vér.-p de repère désactivée
1	1	1	—	X	X	X	—	Vér.-p d'espace désactivée

Page manquante
(publicité)

Page manquante
(publicité)